ED 013 978   24   AA 000 246

THE DEVELOPMENT OF A COMPUTER MODEL OF THE CONCEPT ATTAINMENT PROCESS--A PRELIMINARY REPORT.

BY- BAKER, FRANK B.
WISCONSIN UNIV., MADISON

REPORT NUMBER BR-5-0216-OP-5   PUB DATE   OCT 66
CONTRACT OEC-5-10-154
EDRS PRICE MF-$0.50 HC-$2.92   71P.

DESCRIPTORS- *MODELS, COMPUTERS, *CONCEPTUAL SCHEMES, *COMPUTER PROGRAMS, *COGNITIVE PROCESSES, SIMULATORS, *RETENTION

TO BETTER UNDERSTAND THE COGNITIVE PROCESSES INVOLVED IN THE ATTAINMENT OF CONCEPTS BY HUMANS, A MODEL WAS DEVELOPED BY USE OF COMPUTER TECHNIQUES. THE MODEL WAS BASED UPON THEORETICAL GROUNDS, "THINK ALOUD" PROTOCOLS, AND SPECULATIONS ABOUT THE NATURE OF CONCEPT ATTAINMENT. IT WAS DEVELOPED WITH THREE MAJOR ASPECTS--(1) CONTEXTING FUNCTIONS (HIGHER LEVEL COGNITIVE BEHAVIOR ASSOCIATED WITH SELECTION, MAINTENANCE, AND EVALUATION OF BEHAVIORS), (2) OPERATIONS BEHAVIOR (PERFORMED DURING EXECUTION OF A CONCEPT ATTAINMENT STRATEGY, SUCH AS CREATING SEARCH CRITERION, COMPARING OBJECTS, AND PRESENTING CONCEPTS), AND (3) A MEMORY COMPONENT DESIGNED TO FACILITATE THE OTHER ASPECTS AND FORM A BASIS FOR A MODEL OF HUMAN MEMORY. A CHRONOLOGY OF THE DEVELOPMENT OF THE THREE MAJOR ASPECTS OF THE MODEL IS GIVEN TO PROVIDE INSIGHT INTO THE NUMBER AND NATURE OF THE PROBLEMS INVOLVED IN DEVELOPMENT OF A COMPUTER MODEL OF COGNITIVE PROCESSES. THE CONCLUSIONS WERE SUMMARIZED IN THREE GENERAL AREAS--(1) MODELING CONSIDERATIONS, (2) RESEARCH IDEAS GENERATED BY THE COMPUTER MODEL, AND (3) THE STATE OF THE ART. THE COMPUTER PROGRAM WAS CONSIDERED AS A REPOSITORY OF IDEAS ABOUT THE PROCESSES INVOLVED IN CONCEPT ATTAINMENT, THE IDEAS BEING EXPRESSED IN THE FORM OF COMPUTER PROGRAMS WRITTEN IN COMPUTER LANGUAGE. THE AUTHOR CONCLUDED THAT (1) WHEN COMPARED WITH EARLIER VERSIONS THE CURRENT MODEL SEEMED SOPHISTICATED, BUT WHEN COMPARED WITH HUMAN CONCEPT ATTAINMENT THE MODEL WAS VERY RUDIMENTARY, (2) THE MAJORITY OF INFORMATION PROCESSED BY HUMANS IS INTERNALLY CREATED, (3) THE EXISTING PSYCHOLOGICAL THEORIES AND PUBLISHED RESEARCH DID NOT PROVIDE THE INFORMATION NEEDED FOR FURTHER DEVELOPMENT OF THE MODEL, AND (4) THE PROCESS OF DEVELOPING THE MODEL LED TO IDENTIFICATION OF SOME PROBLEMS THAT NEEDED TO BE SOLVED TO MAKE FURTHER PROGRESS POSSIBLE. (AL)

BR 5-0216
PA 24

OCCASIONAL PAPER NO. 5

[illegible]

Frank B. Baker

RESEARCH AND DEVELOPMENT CENTER FOR LEARNING AND RE-EDUCATION

THE UNIVERSITY OF WISCONSIN
MADISON, WISCONSIN
U.S. OFFICE OF EDUCATION
Center No. C-03 /
Contract OE 5-10-154

Occasional Paper No. 5

THE DEVELOPMENT OF A COMPUTER MODEL OF THE CONCEPT ATTAINMENT PROCESS: A PRELIMINARY REPORT

Frank B. Baker

Associate Professor of Educational Psychology

Research and Development Center
for Learning and Re-education
The University of Wisconsin
Madison, Wisconsin

October 1966

The project development reported herein was performed pursuant to a contract with the United States Office of Education, Department of Health, Fducation, and Welfare, under the provisions of the Cooperative Research Program.

Center No. C–03 / Contract OE 5–10–154

FOREWORD

The R & D Center for Learning and Re-Education has as its primary goal the improvement of cognitive learning in children and adults, commensurate with good personality development. Through synthesizing present knowledge and conducting research to generate new knowledge, we are extending the understanding of human learning and the variables associated with efficiency of school learning. Knowledge is being focused upon the three main problem areas of the Center: developing exemplary instructional systems, refining the science of human behavior and learning as well as the technology of instruction. and inventing new models for school experimentation, development activities, and so on.

As Professor Baker states in the Preface, the computer simulation project described in this report has been in progress since the founding of the R & D Center at Wisconsin in 1964. The project is generating many ideas for gaining knowledge about the psychological processes in concept learning. Research ideas to challenge the most inventive psychological experimenter have been generated. Answers to the first more simple questions have raised additional questions about more complex problems. Although many substantive questions about learning remain to be clarified regarding simulation of concept attainment, rapid progress has been made in computer technology. Sophisticated models and procedures are described in this report.

Herbert J. Klausmeier
Co-Director for Research

PREFACE

The pioneering work at the Carnegie Institute of Technology, lead by A. Newell and H. A. Simon, aroused considerable interest in non-numeric computing; however, the lack of readily available list processing languages limited the number of persons able to engage in this activity. In early 1962, Dr. R. K. Lindsay and J. H. Dauwalder, the University of Texas, programmed IPL-V for the Control Data 1604 computer, thus making IPL-V available to the University of Wisconsin. With this new capability at hand, the present author decided to develop computer programs which simulated some aspect of cognitive behavior. Concept attainment was chosen for a number of reasons, paramount of which was that Dr. H. J. Klausmeier and his students had been working in the area for several years and would provide knowledgeable resource persons. In addition, Hunt's book [1962] was available which provided an entry into unfamiliar literature; and, finally, the concept attainment process looked as if it would be easy to simulate by means of a computer program.

The first program, which I wrote myself in the fall of 1963, served primarily as a device for learning IPL-V. The experience gained from this program convinced me that much could be accomplished and a computer simulation project was written into the original R & D Center proposal. A graduate school research grant during the summer of 1964 supported planning for a long-term project; many of the fundamental ideas were developed that summer. The past two years have been spent in what seems to be an endless loop of running subjects, writing programs, and redesigning the model. Since the initial program, considerable progress has been made; however, we are far from our goal of modeling the processes of human concept attainment.

The purpose of the present occasional paper twofold. First, it is to describe where we currently stand in our research efforts and perhaps provoke some research in the areas we feel are important. Second, it is to present a rather complete history of the development of this project over the past three years in order to provide others with some insight into the nature and magnitude of the problems a neophyte encounters when developing computer models of cognitive behavior.

I would like to emphasize the crucial role played by Mr. Tom Martin, who has programmed all but the first in the long series of programs. He has consistently worked to prevent the programs from becoming what programmers refer to as a "Kludge" and has forced me to sharpen my rather fuzzily conceived ideas. Many of my pet schemes have fallen apart and others have been coalesced into vastly improved schemes by his penetrating inquiries. He has also independently developed programs such as MIMIC which are significant contributions to the programming art themselves.

Mr. Alan Pratt collected the first two sets of protocol data, and Miss Carin Cooper has collected the remaining five sets. Miss Cooper has also thoroughly reviewed literature in simulation and memory thus relieving me of a tedious task.

I hope that this report will be of use to both the psychologist interested in concept attainment and the computer specialist.

F. B. B.

Madison, Wisconsin
1 September 1966

TABLE OF CONTENTS

	page
List of Tables	ix
List of Figures	xi
Abstract	xiii
I. Introduction	1
Goals of the Project	1
The Experimental Situation to be Modeled	3
Development of the Computer Model	3
"Think-Aloud" Protocols	4
Utilizing the Computer Programs	4
Summary	5
II. A Computer Model of the Concept-Attainment Process: CASE Mark IV, Mod 2	6
Introduction	6
Assumptions	6
Representing Cognitive Process	7
Memory Structure	8
The Contexting Hierarchy	8
The Current Program: Mark IV, Mod 2	9
The Structural Details of the Computer Program	14
Symbolic Representation of Behavior	14
Memory Structure Mechanics	15
The Memory Entry Point	18
The Details of the Phase List to the P Level	19
Summary of Chapter	23
III. A Chronology of the Development of the Computer Model	25
Mark I, Mod 0	25
CASE 1, dated 10 August 1964	28
CASE 2, dated 9 November 1964	29
CASE Mark III, Mod 0, dated 3 February 1965	32
CASE Mark III, Mod 1, dated 9 March 1965	36
CASE Mark III, Mod 2, dated November 1965	40
Conservative-Focusing Strategy	40
Wholist Strategy	40
Gambling-Focusing Strategy	41
Conservative-Focusing with Unintentional Variation of Two Dimensions	41
CASE Mark IV, Mod 0, dated 15 December 1965	42

	page
CASE Mark IV, Mod 1, dated 6 January 1966	42
Summary of the Chapter	47
IV. Summary and Conclusions	51
Modeling Considerations	51
Internal vs. External Information	51
The Memory Model	51
Attribute Structure	52
Use of Protocols	52
Contexters	52
Programming Techniques	53
Research Ideas Generated by the Computer Model	54
The State of Our Art	55
References	56
Appendix	57

LIST OF TABLES

Table		page
1	Symbolic Representation of the Conservative-Focusing Strategy List as Used in Mark IV, Mod 2	7
2	Phase Lists to the P-Q Level as Created by the C61 Context Routine	12
3	Symbolic Representation of Routine P61	15
4	Symbolic Representation of the Strategy Used in Mark II	31
5	Symbolic Representation of the Conservative-Focusing Strategy in Mark III, Mod 0	33
6	Symbolic Representation of the Conservative-Focusing Strategy to the P/Q Level for Mark III, Mod 1	37

LIST OF FIGURES

Figures		page
1	One of the 72 Possible Animal Configurations	4
2	Flow Chart of the High Level Contexter List S3	10
3	Partial Representation of the Circular Memory Structure Created by the Computer Program Mark IV, Mod 2	16
4	Typical Modular Memory Module Used in Short-Term Memory	18
5	Flow Chart IPL-V Concept Attainment Program Mark I, Mod 0	27
6	Flow Chart for Mark IV, Mod 1 Containing All Levels of Contexting Routines	47

ABSTRACT

Development of the model described in this report was begun to obtain a better understanding of the psychological processes underlying human concept attainment. The model has been based upon theoretical grounds, "think aloud" protocols, and speculations as to the nature of concept attainment. The model developed is embodied in a computer program written in the IPL-V language. The program exists primarily as a device for expressing complex ideas and relationships in a convenient form. The current version of the program, called Mark IV Mod 2, exhibits a wide range of the behavior observed in the "think aloud" protocols obtained from human subjects.

The model as currently developed consists of three major aspects: contexting, operations, and memory. The contexting aspects of the model are concerned with the higher level cognitive behavior associated with selection of appropriate behavior, maintenance of goal-directedness, and evaluation of completed behaviors. Such functions were labeled contexting as the associated computer programs essentially analyze the current situation and define the context within which the operational routines are executed. The operational aspects of the model are those behaviors which are performed during the execution of a concept attainment strategy. Such behaviors as creating a search criterion, comparing objects, and presenting concepts were considered operational. The memory component of the model was designed to facilitate the other aspects of the model as well as form the basis for a model of human memory. The memory was divided into three types of storage, each used for a particular purpose. The working memory was a buffer-type memory which received information from the external world and acted as a communication device for the transfer of internally created information. The short-term memory contained all of the information relevant to the attainment of a particular concept. The short-term memory was constructed as a "circular memory structure" with a modular format; such a structure enables memory to grow as information is created. The long-term memory will retain learning strategies and descriptive information necessary to implementation of these strategies.

The chronology of the development of the three major aspects of the model has been traced to provide the reader with an insight into the number and nature of the problems facing the developer of a computer model of cognitive processes.

Because existing psychological theories and published research do not provide the types of information necessary to further development of the model, a number of areas of fruitful research have been described. For example, the model suggests that the majority of the information processed by human subjects is internally created; hence new techniques are needed to elicit this information.

When compared to earlier versions, the current model has considerable power and seems sophisticated; however, when compared to human concept attainment it is very rudimentary. Hopefully the current model can serve as the basis for further interesting research.

I
INTRODUCTION

GOALS OF THE PROJECT

The fundamental purpose of the present project has been to obtain a better understanding of the psychological processes involved in the attainment of concepts by humans. The vehicle through which these understandings have been acquired is that of computer programs which serve as models of the concept attainment process. The use of computer programs as models of cognitive behavior has its origins in the early work of Newell, Simon and Shaw [1958] who proposed that the "Logic Theorist" program was a model of human problem-solving behavior. Later work by this group and others has resulted in a well established field which is generally called simulation. But, because the word simulation is widely used in fields other than psychology, most authors currently prefer to use the term "computer models" rather than simulation. The use of computer programs to represent a psychological process involves a number of factors which make the technique extremely valuable. First, because of the small steps by which computers proceed, it is difficult to write programs for something which is ill-defined. Hence, the computer forces one to probe very deeply into a psychological process in order to understand it well enough to program it. Second, the computer program can be manipulated in a number of ways such that it can assist one in understanding the ramifications of the available knowledge about the processes involved. Third, the computer program serves as a repository of the understandings one has acquired up to any given point in time. The ideas are preserved in a language form which is unambiguous and open to study by others. Fourth, in any modeling process one is forced to make assumptions, and in a computer program the role these assumptions play in the model becomes quite apparent. Briefly, the computer serves as an extremely strict task-master who forces one to commit to paper what one understands and, more importantly, what one does not understand.

Because one can approach a problem from many points of view, the emphasis in a project of this type is a function of the interests of the investigator. In that the present author is firmly committed to "process" psychology rather than S-R psychology, no attempt has been made to study the relation of the model to stimulus materials. The interest here is in <u>how</u> a human subject performs the concept-attainment task, not in what variables the experimenter can use to manipulate the subject's responses. Using S-R terms, we are attempting to model the intervening variables, not the gross S-R connections. In the long term an understanding of the internal processes of a human subject holds considerably more promise for yielding new teaching techniques, classroom materials, etc. than does the traditional S-R approach.

At the present time there appear to be two general approaches to the design of a computer model of cognitive behavior, and these shall be referred to below as the "basic premise" approach and the "surface" approach. Under the basic premise approach one postulates a minimum set of operational rules or procedures and then designs a computer program around the successive application of the basic premises or their derivatives to the data presented in the program. The underlying idea is to ascertain how much interesting behavior can be generated by a set of basic premises devised by the investigator. Computer programs which perform pattern recognition [Uhr and Vossler, 1961], the sequence learner due to Simon and Kotovsky [1963], and the concept learning system [Hunt, Marin, and Stone, 1966] are clearly of this type. Such programs assume that a human has the basic premises and the ability to apply them built in or acquired from past experience, and that the investigator has made a reasonable assumption as to what basic premises are involved. An additional, somewhat contradictory, assumption usually involved in such programs is that the program, i.e., subject, begins a given computer run with no past experience relative to the data it will process by means of the basic premises, the "clean-slate" assumption. A considerable amount of intelligence, to use the term loosely, in regard to how to process the data is built into the program, but none in regard to previous presentations of the data is built into the program. The resulting behavior of such a program is typically the construction and modification of decision-trees which are completely dependent upon the sequence of data fed into the program.

The surface approach tends to be associated with computer programs based upon human "think-aloud" protocols [Laughery and Gregg, 1962; Johnson, 1964]. Under this approach one attempts to use the protocols to ascertain the gross behavior patterns of humans in a particular problem-solving or learning situation and then to write computer programs which reproduce these gross behaviors. Such programs can usually reproduce the overt behaviors observed in humans, and the computer-generated protocols can be reasonable facsimiles of corresponding human protocols under the same conditions. In contrast to the basic premise approach, the surface approach does not postulate any specific underlying mechanisms; rather it follows some well-defined, overall plan, such as the concept-attainment strategies of Bruner, Goodnow, and Austin [1956]. In addition, it does not make the clean-slate assumption of the former in that knowledge about the data known to be relevant to a particular phenomenon is built into the program.

In that so little is known about how humans solve concept-attainment problems, making the assumptions necessary for the basic premise approach was not considered appropriate. In addition, the basic premise approach provides very little possibility of discovering new understandings or obtaining new insights as the total system is based upon a preconceived set of basic premises. However, starting under the surface-type approach one can change the system to match the new understandings acquired as one digs further into the problem, rather than being constrained by an artificial set of initial basic premises. Throughout the current project the surface-type approach has been followed, although basic assumptions concerning such things as memory and other facets of this type have been made. However, these assumptions have normally grown out of difficulties encountered within the computer program rather than being preconceived assumptions about the process itself.

In view of the investigators' commitment to the surface approach, a method of attack has been developed which allows one to elicit as much information as possible from the construction of the computer program and at the same time "keep control" of the computer model. The procedure followed is given in the paragraphs below.

One begins with a computer program which corresponds to the behavior of an intelligent subject performing a particular type of concept-attainment problem after having considerable practice. One then slowly builds into the model various types of behavior which are not as efficient as those used by the experienced subject and thus degrades the performance of the computer program. What one attempts to do is work "backwards" toward a computer program which eventually will be as inefficient and stumbling as a person attempting the problem for the first time. By deliberately introducing a particular change into the computer program and then observing how the subject's performance of the task is degraded by that change, one gets an understanding of the ramifications of each change made in the computer model. Such an approach is somewhat at variance with a large number of other simulation projects which have attempted to write a computer program for a subject who is initially very inefficient and inept at solving a problem and then attempted to have the computer program improve its performance to that of an experienced subject. The latter approach appeared to the present author to be a more difficult task as at the current state of knowledge one does not have a particularly good grasp of the causes of inefficient, inept performance. Developing schemes for improving behavior seemed to demand knowledge beyond our current understanding of the situation; it seemed much more appropriate to start from the experienced subject and slowly work backwards with a good understanding of each backward step and its effects upon performance. Thus, the amount of variability in the behavior built into the computer program is a function of our understanding of the concept-attainment process. Eventually the computer program will become as inefficient as a human attempting a problem for the first time, but at that point the reasons for this level of performance and the processes by which a subject improves his performance over a sequence of problems will be understood. At that level of development one would expect to have an extremely good model of the concept-attainment process which could then be used as a guideline for further educational-psychological investigations in the classroom.

The so-called backward approach has proven to be very feasible and quite rewarding in terms of the understandings of the concept-attainment process that have been obtained. The backward approach allows one to continually tie the computer model back to actual subject behavior and to insure that what has been built into the computer program does in some manner represent actual subject behavior. It does not imply that the mechanisms used are true representations of the subject's internal processes; however, the external behavior of the program can be observed in subject behavior.

THE EXPERIMENTAL SITUATION TO BE MODELED

The type of experimental situation for which a computer model is being developed is that reported by Bruner, Goodnow, and Austin [1956] and used extensively by psychologists. The subject is seated before a board containing a number of objects. Each object contains, say, _m_ dimensions and each dimension has _n_ values; thus a complete board has $\underline{n}^{\underline{m}}$ different objects. The experimenter explains to the subject that the objects can be divided into two mutually exclusive groups, members and nonmembers of the set defined by a classification rule (concept). The experimenter designates an object (the focus object) as an exemplar of an object which is a member of the set. The subject's task is to discover the classification rule consisting of a particular combination of dimension values by choosing objects and having the experimenter designate their set membership. When the subject feels he knows the underlying classification rule, he tells it to the experimenter. If the rule is correct, it is assumed that the concept has been attained, and, if not, the subject continues until he can present the correct classification rule.

Bruner, Goodnow, and Austin [1956] identified and labeled a number of strategies which subjects employed in this experimental situation and two, the "conservative focusing" and "wholist," are of interest in the present paper. Using the conservative focusing strategy, a subject chooses an object from the board by selecting an object identical to the focus object, except for one dimension whose value has been varied. If such an object is designated as a member of the set, a yes object, the subject knows the dimension is not included in the classification rule, hence is irrelevant. If the object so chosen is designated as not being a member of the set, a no object, the subject knows the dimension is relevant and the dimension value of the focus object is included in the classification rule. In the conservative focusing (C/F) strategy, the subject varies one dimension at a time and systematically checks each of the _m_ possible dimensions. With this strategy, the minimum number of object choices to attain the concept is _m_, the number of dimensions.

Using the wholist strategy, the subject determines the classification rule through the intersection of all objects designated as members of the set by the experimenter. If an object chosen by the subject is designated by the experimenter as a member of the set, it will have certain dimension values in common with the focus object. Thus, a yes object is of value to the subject and a no object is of no value under this strategy. The subject continues developing the intersection of a series of yes objects and the focus object until he feels he knows the concept. Typically, under this strategy, subject presents a concept for designation by the experimenter after each yes object.

Although both strategies attain the concept, they differ in two major aspects. First, the method for choosing objects under the C/F strategy is well defined and quite obvious to the observer, whereas under the wholist strategy the object choice mechanism is not so clearly observable. Second, the meaning of a yes and no designation of an object choice is reversed in the two strategies. In the C/F strategy a no is the desired designation and in the wholist a yes is the desired designation. Programs for both of these strategies have been developed in the present project, but the primary emphasis has been upon the conservative-focusing strategy.

DEVELOPMENT OF THE COMPUTER MODEL

The books by Bruner, Goodnow, and Austin [1956], Miller, Galanter, and Pribram [1960], and Hunt [1962] and journal articles on the concept-attainment process were read to develop some understanding of what others had done in the concept-attainment area. On the basis of this initial investigation and the author's own intuitive understanding of how he would solve a concept-attainment problem, a computer program which would "simulate" concept-attainment was written. The initial computer program, called Mark I, Mod O, was published in early 1964 [Baker, 1964]. On the basis of this program, protocols were collected to ascertain how sophomores from the University of Wisconsin who had not previously seen this kind of problem would solve it. The "think-aloud" procedure was used to collect data which was then analyzed by the project staff. Analysis of the protocols indicated that the majority of the subjects very rapidly developed a conservative-focusing strategy. Therefore, the computer program was redesigned to incorporate a conservative-focusing strategy. A computer program to model a specific subject has not been developed; rather the "normative" behavior of many subjects, both male and female, was modeled.

"THINK-ALOUD" PROTOCOLS

The data gathering device used throughout the project has been the "think-aloud" protocol as given in the Appendix. As the experiment was being run, the subjects verbalized what they were doing and why they were doing it. Such a procedure has been a standard practice among those developing simulation programs even though it is not held in high esteem in many psychological circles. It was found quite early that the raw protocols were not very rich in information and a modified system in which the experimenter asked pre-planned questions at certain points within the problem was adopted. The questions arose from the computer program and were designed to help fill the gaps in the program. For example, at one point the interest was in whether subjects remembered specific object choices; thus after the fifth object choice, they were asked to identify the second object chosen. Such information would not have been yielded by the usual protocols yet was easily obtainable through selective interrogation. A total of seven sets of protocol-gathering sessions, each involving five male and five female subjects, have been conducted, tape-recorded, and reproduced in mimeographed form. In each of these seven runs a different set of questions was used.

Analysis of the early protocols revealed that the materials used by Bruner et al. [1956] and Klausmeier, Harris, and Wiersma [1964] involved psychologically dependent dimensions. It was found that subjects were unable to treat the dimension of shape independently. To overcome the problem, new materials consisting of animals whose dimensions were ears (long-short), neck (long-short), body (thin-fat), color (yellow-blue-brown), and tail (straight-bent-curly) were devised. Figure 1 presents one of the 72 possible animal configurations. Two of the dimensions were three-valued to overcome the artificiality of all binary-valued

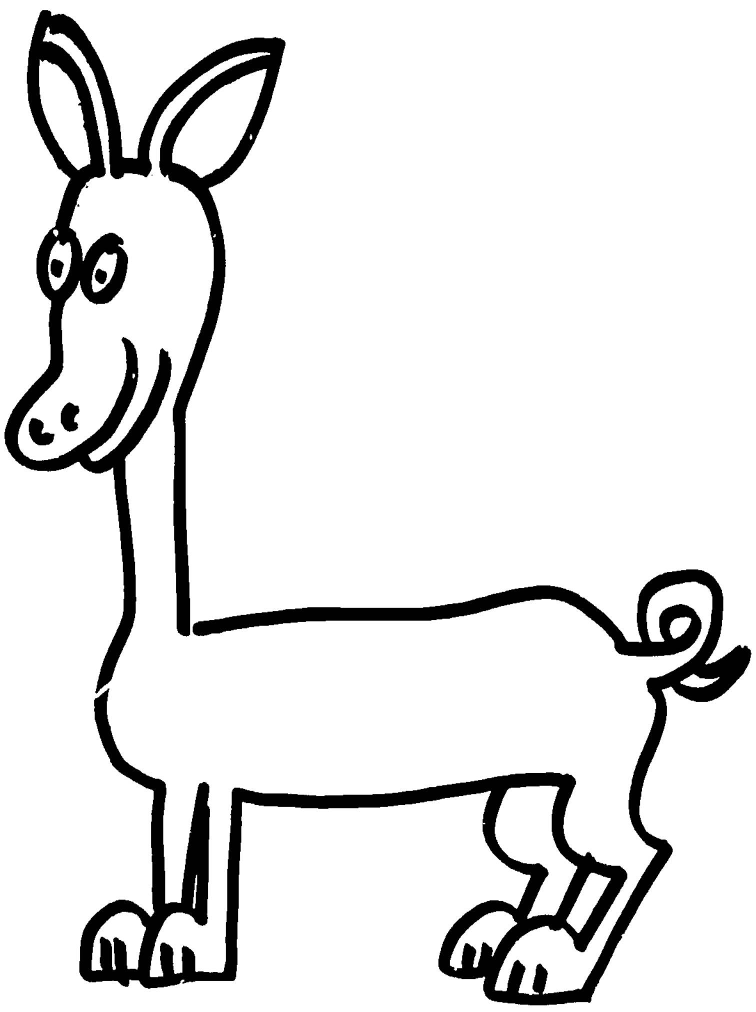
Fig. 1. One of the 72 Possible Animal Configurations.

dimensions. The new materials have proved very successful and will continue to be used in future experiments involving human subjects.

UTILIZING THE COMPUTER PROGRAMS

After writing a computer program to model the behavior of the subjects on the concept-attainment task, one spends a considerable amount of time analyzing the computer program itself in order to reduce it to a simple structure. It is very easy to become trapped with a computer program which is so complex and clumsy that it does not lend itself to the continual modification required by the so-called backward approach. Therefore, extreme care has been exercised to avoid a situation in which one has to periodically start from the beginning.

After a version of the computer program has been reanalyzed, rewritten, and polished to the point where it is a reasonable representation of current understanding of the concept-attainment process, considerable effort is devoted to looking at the points where insufficient information exists. Questions which will help clarify the points of concern are then devised for the next protocol run. Thus, a large feedback loop exists in which attention shifts from subjects to computer program, to subjects and then to the computer program again.

Although it does not show directly in the computer model of the concept-attainment process, a vast amount of effort has been devoted to the mechanics of the computer program itself in order to facilitate the modeling process. Much effort has been devoted to devising data representation schemes, methods of communicating information within the computer program, and methods of executing the computer programs representing various types of behavior. What has evolved is essentially a small computer programming system within which a model of the concept-attainment process is being developed. Strange as it may seem, much of the understanding of the concept-attainment process obtained in this project has arisen out of attempts to develop a systematic computer program for use in the modeling process.

SUMMARY

The goal of the present project is to develop a model of the cognitive processes involved in human concept attainment, and it is toward the understanding of these processes that the above procedures have been directed. To a lesser extent, this project is concerned with developing a computer program which will be an interesting tool in producing further understandings and insights into the concept-attainment process. There has not been interest in developing a computer program which can generate large amounts of interesting behavior whose correspondence to human behavior can be overstated, as has been so typical of past efforts. The computer program is considered to be a repository of ideas about the processes involved in concept attainment, ideas expressed in the form of computer programs in the IPL-V language. Recording one's ideas in this way may seem peculiar, but in a problem as complex as concept attainment, it is virtually impossible to give verbal representation to all of the facets involved.

II
A COMPUTER MODEL OF THE CONCEPT-ATTAINMENT PROCESS
CASE MARK IV MOD 2

INTRODUCTION

The goal of the present chapter is to provide the reader with several levels of description of the most recent model of the concept-attainment process produced by the project staff. One level will be rather gross so that the internal structure of the program can be seen without the clutter of mechanical details. The second will be at the sub-routine level to provide the reader with some appreciation of the formidable problems faced in implementing a computer model of cognitive behavior. In order to present the latter level it is necessary to discuss various mechanical details underlying the actual computer program. A full understanding of the model can only be obtained through a detailed study of the listing of the computer program; however, such an undertaking is beyond the scope of this report. Although the computer program has been written in IPL-V [Newell, 1964], a serious attempt has been made to describe the program without involving more than a bare minimum of the IPL-V language.

Assumptions

A certain number of assumptions were made in order to program the present computer program. The foremost of these is related to the processes of perception which were ignored, even though a large proportion of the errors made in the concept-attainment process can be attributed to perceptual errors of one type or another. The assumption has been made that the subject's perceptual processes are perfect, and they have not been modeled. Secondly, the assumption has been made that memory is perfect; i. e., the computer model does not contain any forgetting processes. At some later point in time, it is anticipated that both decay and interference-type forgetting can be introduced into the computer program, but at the current time such mechanisms would obscure other more crucial aspects.

A major effort in the development of this computer model has been devoted to eliminating the necessity for large numbers of input parameters and prestored information. At the current time only three types of information are prestored for use by the computer program. One of these is the dominant dimension values. Analysis of the initial protocols indicated that subjects possess a preference for certain dimensions and certain values of these dimensions. For example, it was found that female subjects invariably utilize the dimension of color rather early in the solution of their problem and that certain people prefer yellow over blue or brown. Built into the computer program is a selection device based upon probability values assigned to the dimensions and to their values. However, this information is only utilized at one point in the computer program and is not crucially involved in many of the psychological processes. It should be noted, however, that considerable variability in behavior can be accounted for by these dominance values. Three constants which help mechanize certain types of within-problem variability have also been prestored in the program. These constants are associated with the number of dimensions that a subject will use during a particular concept-attainment problem and the number of dimensions he will add to his initial approach when he discovers that it has not worked. The third and final prestored parameter is one known as an awareness factor. The protocols have indicated that many subjects use less than the total number of dimensions in their problem solution and that some of these people are aware of the fact that they are using less, others are not. Therefore, a symbol, or flag, is used to indicate whether the subject is aware that he is using less than the full number of dimensions in his approach to the problem. Other than these three types of information, all data gained by the subject is stored in memory as it is either received from the external world or created by the subject himself.

Representing Cognitive Process

In order to describe the computer model of the concept-attainment process it is necessary to explain a certain amount of symbolic representation used internally by the computer program. In that this project was influenced quite heavily by the earlier work of Bruner et al. [1956] and of Miller, Galanter, and Pribram [1960], the program is built around the idea of strategies, and the mechanics of the program are designed to implement strategies or plans. Quite early in the project it was discovered that one must possess the capability to minimize the impact of significant changes while simultaneously maximizing the ability to make such changes. Therefore, a pseudo-code system and an interpreter, both using IPL-V, were developed to solve this technological problem [Baker and Martin, 1965].

The strategy consists of an IPL-V list of symbols representing routines which are to be performed as well as local symbols indicating branches in the program. These lists, however, do not contain any IPL-V instructions and are not executable IPL-V programs. The list in Table 1 represents a typical concept-learning strategy expressed as a list of symbols.

Each symbol on the list can be the name of a list of symbols, and this representational form can be carried to any depth desired. The symbols are referred to as pseudo-codes as they are merely abstract representations of psychological processes. In the current program there are three levels in the list structure which constitute a learning strategy. The highest level, the S level, is essentially an executive level description of the overall learning strategy. The second level consists of major procedures, the Z and D routines, which perform salient tasks such as hypothesis generation. The third and lowest level are the P's and Q's which are executed to perform the information-processing tasks necessary for concept attainment. The P's and Q's are contained within the Z's and D's and the Z's and D's are contained within the S. Throughout the list structure a distinction is maintained between programs which do things, the Z's and P's, and those which provide decision-making information, the D's and the Q's. The former are analogous to the O routines and the latter to the T routines in Miller's [Miller et al., 1960] TOTE units. Only the lowest level routines can result in the direct execution of subroutines coded in IPL-V, and the higher levels serve only to hold together various combinations of executable routines. The underlying principle is that the P's and Q's are the basic information processing capabilities possessed by a subject, and various tasks are performed by assembling the proper sequence of P's and Q's into Z's or D's. The Z's and D's are then assembled into the strategy list (S). Such a strategy list is then executed by a special purpose interpreter [Baker and Martin, 1965] which works its way through this list structure until it finds a routine which is executable, namely a P or Q routine. It executes the routine and then returns up to the next higher level to ascertain the next executable routine. Fundamentally the interpreter is an ordinary IPL-V recursive program which calls upon itself to

Table 1

Symbolic Representation of the Conservative-Focusing Strategy List as Used in Mark IV, Mod 2

S2	9-0		
	Z0		Process focus information
	C21		Create procedure Z7
	Z7		Establish search criterion
	D4		Determine whether subject should proceed
	9-2		No, error exit
9-1	Z1		Construct search criterion
	Z2		Select object from external environment
	C37		Create decision procedure D1
	DO		Determine whether object selected meets subject's needs
	9-1		No
	Z3		Experimenter designates set membership of the object choice
	C38		Create routine Z4
	Z4		Process information gained through object designation
	D1		Determine whether a concept can be presented
	9-1		No
	Z5		Form a concept
	Z8		Experimenter designation of correctness of concept
	C22		Construct procedure Z6
	Z6		Subject's reaction to designation of concept
	D3		Determine correctness of concept
	9-1		No
	0	0	Yes
9-2	X21		Error exit
	0	0	

work its way up and down the branches of the list structure representing the learning processes.

Memory Structure

Quite early in the development of the concept-attainment program, it was determined that memory plays a crucial role in the concept-attainment process, and it was necessary to design a rudimentary model of memory. In the model, memory is divided into three major aspects: working memory (WM), a temporary, buffer-type memory; short-term memory (STM) in which all information relative to a given problem solution is stored; and long-term memory (LTM) in which the subject stores information to be retained over longer periods of time. Thus, the breakdown of memory is a function of the duration of time over which the information is to be retained. Such a three-part breakdown of memory does not correspond directly to the memory model ordinarily used by psychologists which involves only a short-term memory and a long-term memory. Most of the functions of this project's short-term memory are embodied in their long-term memory. However, investigation of the protocols seems to indicate that subjects retain information about a problem only long enough to solve that particular problem and then do some recoding to save the salient features over longer periods of time. Therefore, it was suggested that there is a distinction between short-term and long-term memory which psychologists do not normally recognize.

The short-term and long-term memories have a highly interconnected net structure which is developed by the program as information is acquired. The dynamic nature of the memory structure is an important feature of the three level model and is discussed in a later section in the present paper.

The Contexting Hierarchy

The internal organization of computer models constructed under either the basic premise or the surface approach is focused upon implementing a rather specific psychological phenomenon and does not take into direct account a higher level of cognitive behavior, namely that which in some sense directs, maintains, and evaluates the overall problem-solving or learning behavior of a human subject.

In order to clarify this issue, let us briefly examine a problem-solving or learning experiment as it is usually conducted. In such an experiment there is a fairly typical sequence of events which transpire in roughly the following order:

(a) The experimenter explains the nature of the task, the characteristics of the experimental materials, and the types of products the subject is to produce.

(b) The subject relates the given information to what he already knows.

(c) Once the subject has assimilated the information to his own satisfaction, he embarks upon an approach to the task which is resplendent with errors and inappropriate decisions, but, none the less, he exhibits goal-directed behavior.

(d) The subject is able to evaluate, in some sense, how well he is doing by means of both internal and external clues.

(e) With sufficient experience on the same task, the subject is usually able to modify his own behavior to the point where he becomes proficient at the task and his once clumsy performance becomes smooth and effortless.

In that such a pattern of behavior is essentially independent of the particular task, it is very difficult, for the present author at least, to conceive a realistic model of human behavior whose internal organization does not provide for some form of a central executive to account for this communality. The relevant issue is the form of this central executive and the internal organization of a computer program necessary for its representation in a computer model of cognitive behavior. Unfortunately, it is extremely difficult to obtain direct evidence from either protocols or psychological experiments upon which to develop a model of such a central executive. In addition, the method of creating one is not obvious; as Newell [1962] said, "In attempting to create such a central organization we found—as we had in the problem of communicating strategies—that we had no concepts and no formal language to discuss the variety of results and their uses." (p. 410)

In earlier editions of the program the central executive was confounded with the strategy list; however, in the current version, the supervisory or executive aspects of the program have been separated from those of the operational aspects. C routines developed to represent this executive function constitute a hierarchy of control whose role changes as a function of the stage of the task performance. Because the function of the supervisory program changes often, the term "contexter" is probably more appropriate for these routines than "central executive" which carries an unwarranted connotation of a single supervisory program.

Although the role of a contexter is a function of the situation in which it operates, there is nonetheless an underlying communality throughout all levels of contexters which can be described by a series of questions which a context routine attempts to answer: (a) What is the current situation ? (b) What does it mean ? (c) What could be done ? and (d) What will be done ? Thus, whether the contexter is dealing with a gross overall plan of approach to a task, or with some small operation in a subtask, its fundamental framework is invariant; what varies is the situation in which the contexter occurs and the procedures by which it attempts to answer these questions. It is worth noting that the definition of the current situation includes not only the available data but also the sequence of behaviors leading up to the present point in time. The final result of a contexter routine is some executable behavior for which an appropriate context has been established.

The contexter may be viewed as creating a plan or a strategy for behavior. At high levels in the model, it creates a plan for overall behavior such as the S list and at low levels it creates plans for very specific actions such as P lists. Such a planning hierarchy was first envisioned by Miller, Galanter, and Pribram [1960] when they suggested the existence of plans which create plans. Because of the rather complex interrelationship between the contexting programs and the strategy lists, we shall defer a detailed discussion of these to a description of the actual computer program itself.

THE CURRENT PROGRAM–MARK IV, MOD 2

The preceding paragraphs have been devoted to acquainting the reader with some of the major considerations in the design of the simulation program, but let us now turn our attention to the current version of the computer model of concept attainment. The overall picture is as follows: First, the experimenter verbally describes to the subject what the experiment is about, what the board looks like, the dimensions on the board and their values, and in essence describes the experimental situation. The experimenter also indicates to the subject that he is to select objects whose set membership will be designated by the experimenter. When the subject feels he understands the concept, he is to present it to the experimenter for designation. Upon receipt of the instructions, the subject proceeds to try to attain the concept. The computer program begins with an initialization phase which utilizes the subject's past experience and his characteristics, namely some of the constants mentioned earlier and the dominance values, to establish an initial set of conditions within the subject. After completing this initialization phase, the computer program creates a search criterion and locates an object in the external environment which it also feels is a member of the set. If the object found meets the requirements of the subject's search criterion, it is then presented to the experimenter for designation. After receiving the designation of the object, the subject processes the meaning of yes or no in light of his own understanding of the problem. If the subject feels he can present a concept he proceeds. However, in most cases the subject makes several object choices before he has enough information available to decide whether or not he understands the concept. Therefore, at this stage, the computer program creates a new search criterion and locates other objects from the board visible to the subject. The final phase of the program occurs when the subject feels he has enough information to present the experimenter with a concept. If it is incorrect, the subject then has to construct a reaction to this designation and return to the first phase in which he searches for additional objects that will enable him to ascertain the correct concept. If the concept is correct, the problem is terminated and the subject then tries to evaluate what he has accomplished during the problem. So much for an overview of the concept-attainment process. Let us now turn our attention to a discussion of the flow chart which is given in Figure 2 below.

This flow chart will be discussed in terms of the particular routines which it contains. No attempt to go into all the programming or mechanical details is made, but a verbal description of what occurs within the program is given, and any relevant assumptions made by the program are indicated. The highest level program list in the computer model is S3, the high level contexting list. This program essentially contains a gross description of what is to occur in the attainment of a concept and consists of four computer programs: C11, which creates a tentative strategy from the experimenter's instruction; E95 in which the experimenter presents the focus object to the subject; C61 which creates and executes a strategy phase-by-phase until the concept is finally attained; and C12, a problem solution post-mortem analysis in which the subject ties together what he has done into a workable learning strategy for future use. It should be

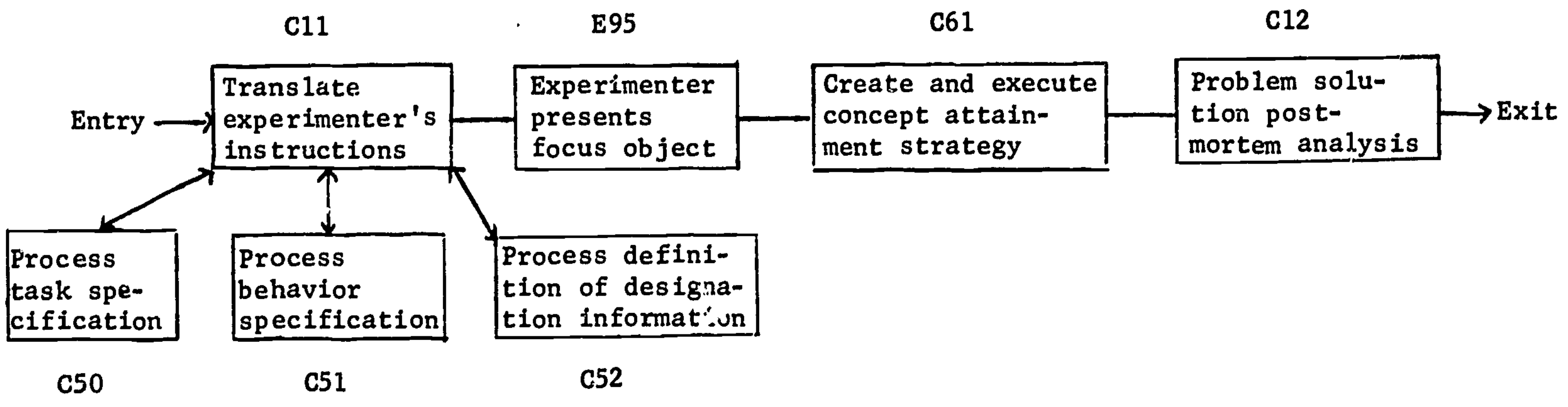
C11
E95
C61
C12
Entry
Translate experimenter's instructions
Experimenter presents focus object
Create and execute concept attainment strategy
Problem solution post-mortem analysis
Exit
Process task specification
Process behavior specification
Process definition of designation information
C50
C51
C52

Fig. 2. Flow Chart of the High Level Contexter List S3

noted that C11, C61, and C12 are contexting-type routines.

The messages from the experimenter explaining the problem to be solved have been coded in terms of attributes and particular attribute values which essentially describe major behaviors; such coding avoids the syntactical-semantic analysis problem normally associated with translating English language into a computer program. Up to the current time this problem has been by-passed completely because it is a major research project in itself. The experimenter messages describing behavior in terms of attributes and values permit a search of long-term memory to see whether other similar behaviors carrying this description are available for assembling into a strategy. Such an approach is rather crude, but it enables one to introduce into the model some forms of translation of instructions to behavior.

The C11 routine accomplishes the translation from experimenter messages to the description of a rough skeleton strategy. In the first problem attempted by a subject, the C11 routine creates a skeleton strategy list which describes the gross behaviors necessary to attack this concept-attainment problem. In subsequent problems, C11 searches the long-term memory for a strategy list from a previous problem that can be utilized as the initial approach to the problem. C11 consists of three major sub-routines, each associated with a different type of message from the experimenter. The first sub-routine, called C50, creates a problem list and indicates what the problem is; in other words it stores the information that the problem is conjunctive and that it is the first problem. The second sub-routine, called C51, creates a description of a skeleton strategy for solving the problem. The skeleton strategy does not include all of the behaviors necessary to attain the concept, but stores the major framework of the experimental situation contained in the experimenter's messages. The third major sub-routine, C52, is designed to store specific types of information which the experimenter presents to the subject, such as that he will designate set membership of an object by the words yes or no. Thus, what C11 does is take in a particular message which corresponds to a sentence or a series of sentences in the experimenter's verbal instructions and translate it into descriptions of particular behaviors which the subject must perform in order to attain a concept. Selection of the specific C50 routine used is a function of the message that has been received from the experimenter, and there are decision processes within C11 enabling the program to call the proper sub-routine for a given message. There are two major outputs of the C11 routine. One is the beginning of the short-term memory structure which the program will grow during its solution of the concept-attainment problem. The initial point of the short-term memory, the symbol L100, represents the problem and carries the description of it obtained from the experimenter messages. The second is a skeleton strategy list containing symbols representing the major procedures within the concept-attainment task as indicated by the experimenter. These symbols are not executable routines at this point, but merely hold descriptions of the kinds of behaviors necessary to accomplish the task. If the subject has previously obtained a concept, rather than construct a skeleton strategy C11 locates the recently used strategy in long-term memory and places its name in working memory.

When C11 has been completed, the subject knows in a general way how he will perform the concept-attainment task. To specify a particular concept-attainment task, the experimenter must identify the focus object, which is an exemplar of an object belonging to the set defined by the unknown classification rule. Routine E95 performs this function by placing the name of the focus object in the subject's working memory. The name of the focus object is accompanied by descriptive information indicating the set membership of the focus object. Upon completion of E95, control of the program returns to the subject.

The major contexting routine in the current program follows E95 and is called C61. The phase lists of the strategy produced by C61 are presented in Table 2. At the time this routine is executed, the short-term memory contains the symbol L100 representing the problem to be solved, and the contents of the working memory are either the name of the skeleton strategy or a previously developed strategy. The routine checks a flag to determine whether this is the first problem it has solved or not. If it is not the first problem, C61 assumes that the contents of working memory are a fully developed strategy which is given to the interpreter for execution. If it is the first problem, the program must then translate the description of the skeleton strategy into executable behavior. The C61 routine creates a symbol for the strategy list and then creates a symbol for the first phase of the strategy. Having created a phase symbol, it searches long-term memory to find a routine whose description matches that of the first routine on the skeleton strategy list. It will be routine Z0 which receives the focus object and its designation from the experimenter and stores them on the problem list. Because Z0 receives information from the external environment, it is followed by a contexting routine. (It should be noted that one of the rules of the computer program is that a contexting routine must follow information received from the external world.) In this case contexter C21 is inserted on the phase list after routine Z0. The contexter C21 creates routine Z7 which uses the focus object and the characteristics of the subject to establish the initial working hypothesis. The working hypothesis serves as the basis for search criteria through which objects are chosen from the board. Because C21 creates routine Z7, a symbol for the latter appears on the phase list produced by C61. This completes the phase 1 list which is then given by C61 to the interpreter for execution. The remainder of routine C61 is a generalized program for, first, creating phases determining whether contexters are required or not and, then, executing the phases.

Phase two is the object-choice phase, consisting of routine Z1 which creates a search criterion by varying one or more dimension values of the working hypothesis, routine Z2 which chooses an object from the board, C37 which establishes the test conditions for routine D0, and the latter which ascertains whether or not the object choice meets the subject's needs. If it does, C61 continues to phase three; if not, it returns to phase two and re-executes it.

Phase three is the experimenter designation of the object phase containing routine Z3, which presents the object to the experimenter for designation, and routine C38, a contexter that establishes routine Z4. The procedure Z4 processes the information provided by the experimenter's designation of the set membership of the object chosen. Following the logic of the conservative-focusing strategy, Z4 flags the dimension or the dimension value as relevant or irrelevant, depending upon whether the object was designated a yes or no. The last routine in phase three is D1 which ascertains whether or not a concept can be presented. D1 checks each of the dimensions of the working hypothesis and determines whether the subject considers them relevant, irrelevant, or untested. If all dimensions have been flagged by Z4 as either relevant or irrelevant, sufficient information is available for the subject to present a concept to the experimenter. D1 may also be conditioned when certain dimensions are still untested if the subject is using less than the total number of the dimensions and has flagged all those he is using. If a concept can be presented, phase four is entered. If not, the program returns to phase two and executes phases two and three over again.

Phase four, which is the final phase, consists of procedures Z5, Z8, C22, Z6, and D3. In routine Z5 the subject searches the dimension values of the working hypothesis for those values which are relevant and from the relevant dimension values creates a concept, i.e., a list of dimension values which it believes define the classification rule. The next routine is Z8 which presents this concept to the experimenter for his designation. Upon designation, a contexting routine, C22, is executed because Z8 brought in information from the external world. C22 is very similar to C38 in that it will create a situationally dependent routine Z6 for utilizing the information provided by the designation of a concept; Z6 is created only if the concept is incorrect as a subject then has to ascertain what has been wrong with his behavior. At the current time, Z6 essentially

Table 2

Phase Lists to the P-Q Level as Created by the C61 Context Routine

Ø1	9-0			
	Z0	9-1		
		P21		Copy focus object
		P61		Remember name of focus object
		C31		Put name of focus in memory entry point
		P62		Remember set membership
		0	0	
	C21	9-1		
		C20		Create Z7
		0	0	
	Z7*	9-1		
		P191		Construct working hypothesis
		P63		Remember name of hypothesis
		C31		Put name in memory entry point
		P64		Remember how hypotheses formed
		0	0	
	D4	9-1		
		D40		Determine whether subject should proceed
		0	0	
Ø2	9-0			
	Z1	9-1		
		P131		Select dimensions to vary
		P141		Select new dimension values
		P151		Create search criterion
		P64		Remember search criterion
		0	0	
	Z2	9-1		
		P51		Search board for object
		P65		Remember object
		C31		Put name of object in memory entry point
		P66		Remember how object found
		0	0	
	C37	9-1		
		C36		Create procedure D0
		0	0	
	DO*	9-1		
		Q50		Determine whether object meets subject's needs
		0	0	
Ø3	9-0			
	Z3	9-1		
		P71		Transfer object to experimenter
		E93		Designate set membership of object
		P62		Remember object designation
		0	0	
	C38	9-1		
		C39		Create procedure Z4
		0	0	
	Z4*	9-1		
		P501		Recall object designation

Table 2 (continued)

		C41	Pop memory entry point list
		P91	Mark relevancy of dimensions
		P171	Revert dimension values
		0 0	
	D1	9-1	
		Q101	Determine whether concept can be presented
		0 0	
∅4	9-0		
	Z5	9-1	
		P121	Form a concept
		P67	Remember concept
		C31	Put name of concept in memory entry point
		P68	Remember how concept formed
		0 0	
	Z8	9-1	
		P72	Transfer concept to experimenter
		E94	Designate correctness of concept presented
		P69	Remember designation of concept
		0 0	
	C22	9-1	
		C23	Construct procedure Z6
		0 0	
	Z6*	9-1	
		Q41	Acquire untested dimension
		P181	Add dimension to working hypothesis
		C31	Put name of hypothesis in memory entry point
			Remember how hypothesis formed
		0 0	
	D3	9-1	
		Q31	Determine whether problem completed
		0 0	

*Routines created at execution time by the preceding context routine.

looks for dimensions which have not been involved in the concept itself. In other words, it looks through the dimensions and dimension values of the focus object searching for untested dimensions. If it finds untested dimensions, it adds them to the working hypothesis list. For example, if a subject initially used only three out of the five possible dimensions, Z6 will add one or more dimensions as a function of the number of available untested dimensions and the value of the parameter K97 which specifies how many dimensions are to be added. If Z6 discovers that all the dimensions have been flagged and the subject still has not attained the concept, it assumes that he has mis-flagged a dimension. Instead of adding untested dimensions to the working hypothesis, it unmarks dimensions on the working hypothesis list so that a new search criterion will include those which have been used in the past. The last routine in phase four is D3 which ascertains whether or not the subject should continue to attempt the problem.

Upon the completion of phase four, C61 realizes it has a list of executable routines for all the behaviors from creating a search criterion to testing the concept; therefore, it treats this list as a sub-strategy; i.e., it is a complete strategy except for the initialization phase, but, because it does not need to reinitialize anything, it can be executed as if it were a total strategy. Hence, if the concept is incorrect, the sub-strategy is executed until the concept is attained.

All four phases created and executed by C61 from the skeleton strategy list have the same general structure. There are one or more major procedures at the Z level. When information is received from the external world, a contexter routine will create a situationally dependent routine to determine the meaning of the external information. The final routine in each phase is a D routine which asks whether the subject can proceed or whether he must return to the object selection phase to get further information.

In review, C61 proceeds step by step and tries to perform the behaviors indicated by the skeleton strategy as in the concept-attainment process. It picks its routines from long-term memory by comparing the description of what needs to be done with the description of the capabilities of routines stored in long-term memory. Phases are created to handle logical units of behavior within the concept-attainment process, and these phases are then given to the interpreter for execution. If progress can be made, C61 will move on to the next phase within the problem, repeating this process until it can present a concept. If, upon completion of phase four, the concept is correct, the program is terminated; if not, C61 merely executes what it has already created as a strategy until a concept is attained.

The routine following the successful completion of a strategy is a post-mortem analysis routine called C12. Because the model has not progressed beyond the within-problem analysis stage, this aspect of the process has not received more than cursory attention. At the current time the task of C12 is to tie together the total strategy which has been created rather piecemeal by C61. It places symbols representing phases one, two, three, and four in a common strategy list and puts links from each of the phases to phase two. At a later date in the project, it is intended that C12 will do an analysis of the execution of the program to ascertain whether there are unnecessary behaviors and to smooth out a successful strategy. C12 also stores the successful strategy on the long-term memory so that it can be used by C11 when a subsequent problem is attempted.

The computer program described above is an attempt to model the salient features of a subject performing a concept-attainment task. The initial stages are quite slow because experimenter instructions must be understood, a rough idea of the procedure must be constructed, and the subject must proceed step by step. Once the full process has been carried out, the pace quickens since the subject repeats behaviors established during the early phases, thus eliminating most of the high level contexting previously required. Hopefully what has been developed is a reasonable framework within which one can continue to investigate the concept-attainment process.

THE STRUCTURAL DETAILS OF THE COMPUTER PROGRAM

Symbolic Representation of Behavior

In order to present a detailed discussion of how the computer program attains concepts, it is necessary to elaborate further upon the internal structure of the program. Attention will be given to the representational scheme for subroutines, the attribute system, and the memory structure.

Let us examine a particular process within the strategy list, say routine P61 which appears in procedure Z0. The list of symbols representing routine P61 is given in Table 3.

The symbol P61 represents a non-executable routine whose function is to hold the description of the executable routine, P60 in this case. The P61 symbol is a pseudo-code defining the context within which the executable routine will function. Thus, a given executable routine may appear on several different pseudo-code lists. Such a feature permits the development of powerful generalized routines which are independent of a particular context. The description list 9-0 of the pseudo-code P61 contains attribute A1 whose value V1 is a list of the inputs to P60. The attribute A2 has on its value list V2, the names of the locations at which the outputs will be placed. The attribute A3 has the symbol A305 on its value list which describes P60 as a routine involving remembering. The descriptions held by the pseudo-code can be used by higher level context routines to ascertain the characteristics of the routine. Such a system provides a rudimentary description of behavior. All routines—whether they are contexters, strategies, procedures, or processes—are represented in the computer program using this scheme.

The special interpreter [Baker and Martin, 1965] extracts the inputs from the list and places them in the IPL-V Communication Cell H0, then executes routine P60 in IPL-V. The outputs created by P60 are left in the location named on the output list of P61. Except for the memory processes, all P and Q level routines leave their outputs in working memory.

The pseudo-code and interpreter system permit one to develop general purpose programs which can be used in a number of different

Table 3

Symbolic Representation of Routine P61

P61	9-0		Pseudo Code	
	P60	0	Executable Routine	
9-0	0			
	A1	Input Attribute		
	V1	0		
		M1	Working Memory	
		F3	0	M1, N Flag
	A2	Output Attribute		
	V2	0		
		M10	Memory Entry Point	
		A20	0	Focus object Attribute
	A3	Process Description Attribute		
	V3	0		
		A305	0	Remembering

situations. For example, processes P61, P62, P63, P64, etc. contain the same executable routine P60 and differ only in the information contained on their respective input-output lists. Thus P61 may store the focus object in short-term memory, whereas P62 may remember the experimenter designation of an object choice. Such a scheme was designed to permit eventual development of contexter routines which will place information on the input-output lists of a pseudo-code rather than having the human programmer code in the information. Thus, it is a step toward programs which can create programs.

Memory Structure Mechanics

Much of the design of the computer model is dependent upon the mechanics of the three-level model of memory employed. In the paragraphs below the working memory and short-term memory are examined in detail.

The working memory consists of only two cells, M1-N and M1-D which are on a list called M1. M1-N contains the name of a particular piece of information, for example, the name of an object chosen from the external world or the name of a search criterion by which the subject is scanning for objects. M1-D contains what has been called an unattached description list or a dummy description list, in the notation normally a DDL. The dummy description list contains a description of the symbol within the M1-N portion of the short-term memory, the idea being that subject has not attached the description to the item itself, but rather has created a description which later routines will process and attach either to the element named in M1-N or to some other item of information. The rationale is that the dummy description list corresponds roughly to a chunk as discussed by Miller et al. [1960]. However, this chunk will not necessarily be attached to the item named in M1-N. The cells M1-N and M1-D merely contain the name of a list and therefore working memory is only two storage elements deep.

The working memory plays two roles within the simulation program. In the first role, it acts as an input buffer from the external world. All information from the experimenter himself, such as the focus object or the designation of an object choice or a concept, comes to the subject through the working memory. For example, in the case of the focus object, M1-N would contain a symbol representing the focus object and M1-D would contain a dummy description list which designates this object as a member of the set. The information left in working memory is then acquired by a subsequent processing routine and can be stored or processed further within the concept-attainment

```
L100 9-0
     X1   0
9-0  0
     A300 External environment
     V300 0
          E9       E9  9-1  Subjects internal M13  9-2   Structure of the external environment
                       Q1   representations         E10      Color   E10 9-3
                       Q2   of the external         E20      Ears        E11        Yellow   E11 9-4  0
                       Q3   environment             E30      Neck        E12        Blue     9-4 0
                       .                           E40   0  Tail        E13     0  Brown        A77
                       .                      9-2  0                9-3 0                       V77
                       .                           E10                  E11                     3624 0
                       Q72  0                      N10 + 50             N11 + 50
                   9-1 0                           E20                  E12        Dominance
                       A101                        N20 + 60 Dominance   N12 + 50
                       V101 0                      E30      Values      E13        Values
                            M13 0                  N30 + 10             N13 + 25
                                                   E40
     A302 Strategy S2  9-1  Strategy list          N40 + 35
     V302 0            Z0
          S2   0       .
      .                .
                       .
      .                D3
                       0    0
      .            9-1 0
                       A18  Search criterion
                       V18  0
                            3671              3671 9-2      Search criterion
                            2033                   E11
                            0112   0               E32
                                                   E41      0
                                              9-2  0
                                                   A16      Objects found
                                                   V16      0
                                                            C10         C10 9-3 Object 10
                                                            C27             E11
                                                            C11      0      E22
                                                                            E32
                                                                            E42 0
                                                                            9-3 0    How object found
                                                                                A26
                                                                                V26  0
                                                                                     2163    2163  9-4 0    "From-to list"
                                                                                     3011 0  9-4   0
                                                                                                   A9       Dimension varied
                                                                                                   V9  0
                                                                                                       E10  Color  E10 0 Color
                                                                                                   A10      from
                                                                                                   V10 0
                                                                                                       E12  Blue   E12 0 Blue
                                                                                                   A11      to
                                                                                                   V11 0
                                                                                                       E11  Yellow E11 0 Yellow
```

Fig. 3. Partial Representation of the Circular Memory Structure Created by the Computer Program Mark IV, Mod 2.

processes themselves. The second role of the working memory is to act as a communication device between various sub-routines within the computer program. In the early days of the present computer program, it was felt that most of the information processed was obtained from the external world. However, protocol analysis very quickly showed that a major portion of the information processed by the subject was created internally, therefore a requirement existed for some means of temporarily storing a piece of information so that a series of processes could work upon it. Most of the low level routines within the computer program receive their information from the working memory and, after processing it, leave their outputs in working memory. In many cases the subsequent routine remembers the information in either long-term or short-term memory. From a programming point of view, the working memory, acting as an internal information buffer, solves many mechanical programming problems which otherwise would become enmeshed in the idiosyncracies of IPL-V itself. Working memory is very similar to the H0 Communication Cell of the IPL-V except that it is in the program rather than the programming language.

The second major portion of the memory structure is short-term memory which contains all of the information relevant to solving a particular problem. The current structure of this memory is one that can best be described as a highly interconnected net. The short-term memory structure shown in Figure 3 is designed to grow as the information in the problem is acquired. However, the growth is constrained by a modular memory structure as shown in Figure 4. For example, in Figure 3 the symbol L100 represents the problem which is currently being solved. On the description of L100 is an attribute A302 whose value describes the problem by means of the current strategy. Describing the current strategy is an attribute, A18, indicating the objects which have been found. Describing the objects which have been found are various "from-to lists" containing the dimensions that had been varied in order to find each object. The memory structure is, in reality, a tree; however, the information on one branch of the tree is not unique to that particular branch. For example, the "from-to list" describing what dimension was varied to find a particular object is also contained on the branch which describes how the search criterion was created from the focus object. The highly connected net of memory was originally conceived as a circular memory; once it has grown over a period of time, there is no real beginning and no real end to it because information is cross-linked and inter-linked so heavily that the structure of the tree has become obscured. At the present time, because of the single problem solution involved, the tree does not get overly complex.

For clarity the symbols in Figures 3 and 4 are IPL-V regional symbols, but in the actual program these symbols are created by the program as the information is acquired. The memory net given in Figure 3 does not exist in short-term memory prior to execution of the program. The computer program possesses the capability to create memory only as it needs to store information. Such a memory capability differs considerably from that usually employed in computer programming in which the programmer accounts for every memory location used. The dynamic memory structure originating here represents a first step toward a computer program which can store and recall information without outside intervention.

Various attributes under which information is stored constitute some of the basic assumptions of the current computer program. These attributes are felt to be an intermediary step between current status and where one wants to be, in that the investigators understand neither how to describe behavior nor how people store information in memory. Therefore, this is an approach to these particularly difficult problems. In order for the memory recall processes to work, a structure was developed which enables the program to tell when it is finally to a point at which information is available; hence a distinction between class and specific attributes was devised.

In Figure 4 the attribute A21 is a class attribute as its function is to hold a series of specific descriptions on its value list V21. The symbols on the body of list V21 are dummies whose sole function is to hold a description list containing specific information. Thus, the description of list Y2 contains specific attributes A4 and A5 whose values are lists V4 and V5. Such a memory structure is symmetrical above and below the dashed line, thus permitting a single set of remember and recall routines to function at all levels. It should be noted that usable information can only be obtained from the specific attribute level; all higher levels are merely symbols representing larger units of information. The class attribute value list V21 is also time ordered with the most recent data at the top of the list. The description lists contained to the left of the brackets describe a particular list and are themselves

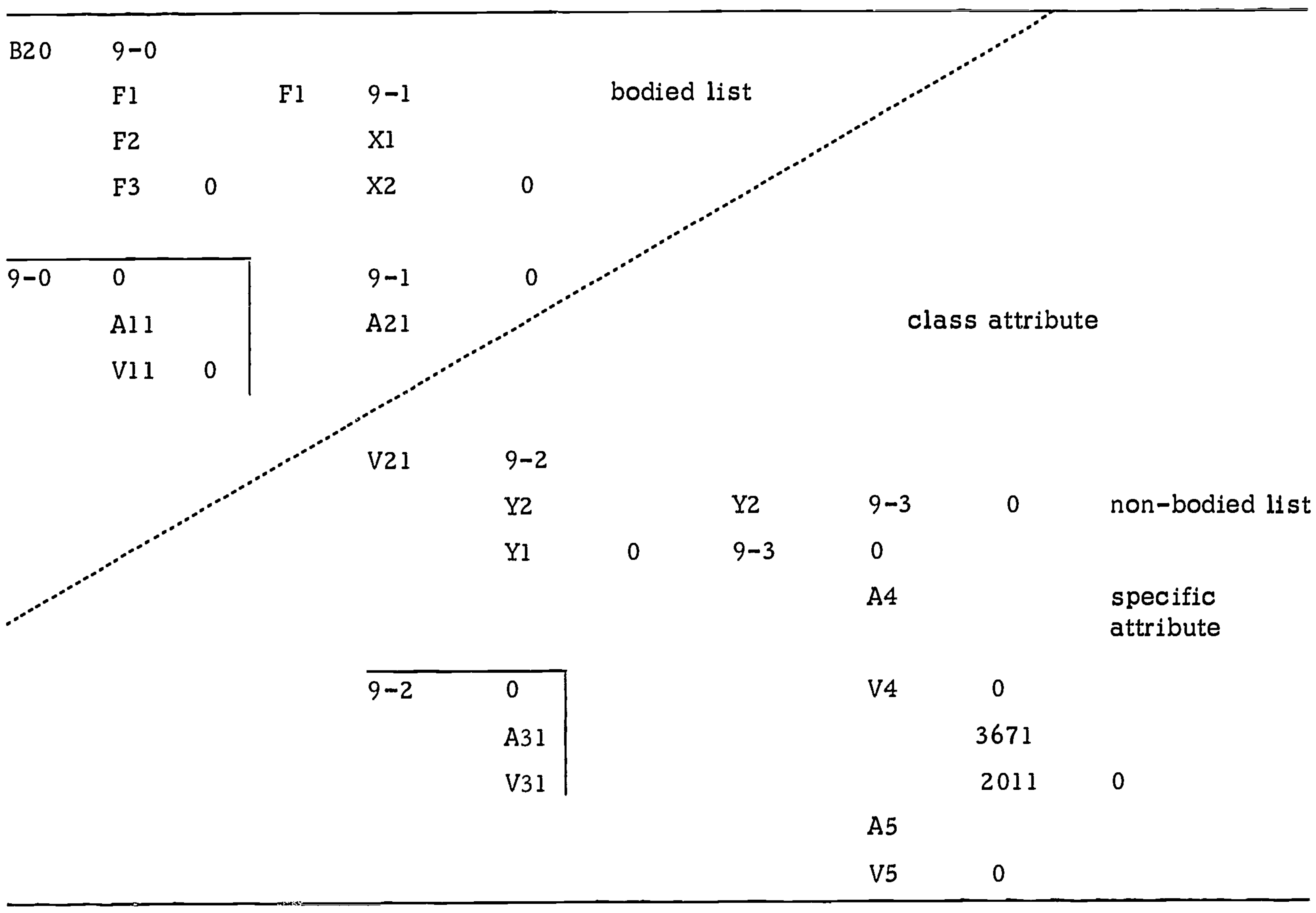

Fig. 4. Typical Modular Memory Module Used in Short-Term Memory

of the same modular structure.

The long-term memory has not been designed because the between-problem variability stage of the project has not been reached. From the initial protocols and from the current computer program, it appears that strategies and key pieces of information necessary to execute a particular strategy are stored in long-term memory. It does not appear that a great wealth of detailed information is ever stored in long-term memory. A future stage of development of the current program will be devoted to studying the problem of long-term memory and trying to realize an adequate model for this aspect of the memory structure.

The Memory Entry Point

One of the major problems faced when developing the circular memory structure was some means for entering the memory or at least keeping track of the location in memory having entered it. The device invented for the latter purpose was called the "memory entry point." As one looks at the concept-attainment process, it becomes evident that, as the subject goes through the various phases, the information created is normally about a particular point within the process. For example, when an object is chosen from the external environment, the subject spends a fair amount of time processing various types of information about this object, what dimension was varied, what the experimenter's designation of the object was, etc. Much of the information to be stored or recalled is related to the particular item. Therefore, the object chosen serves as the memory entry point. As the process moves on to another piece of information, for example, the creation of a concept, the memory entry point changes. However, this change is normally either upward or downward on the branch of the short-term memory so that the memory entry point is really a push-down, pop-up list in which the subject keeps track of where he has been in memory. The problem of how to initially enter the memory structure has not been resolved, but once in the memory structure the computer program can keep a record of where it has been. Because several of the context routines have to revert to previous levels, there are two small routines, C30 and

C40, which add names or take them away from the memory entry point list. The memory entry point technique is not a very satisfactory solution; however at the present time it is a feasible one to program until we develop a better understanding of memory processes.

The remembering of information in short-term memory and the recalling of information from it are accomplished by generalized processes P60 and P500, respectively. The pseudo-code, say P61 as in Table 3, containing P60 has on its input list the symbol representing working memory (M1) and a flag indicating whether M1-N or M1-D is to be remembered. The output list of the pseudo-code contains the symbol representing the memory entry point (M10) and the descriptive attribute (A20) under which the information is to be stored. All storage operations are assumed to describe the symbol named in the memory entry point, and the basic process is unaffected by the type of information stored. The distinction between storage under a class attribute and storage under a specific attribute is handled within the basic P60 routine, thus the program does not need to concern itself with this distinction. The basic recall routine is P500 which is the converse of P60 and shares much of its internal programming.

The communication of the subject to the experimenter is one of the aspects of the concept-attainment process of little concern from a psychological point of view. Therefore, all communications to the experimenter take place through a special output register called E1 into which the subject puts information and from which the experimenter removes information in order to designate objects or concepts. It is mechanically simple, but not necessarily psychologically sophisticated.

THE DETAILS OF PHASE LISTS TO THE P LEVEL

Having described some of the underlying mechanics, let us turn our attention to the details of the phases created by routine C61. So that the reader may get the "flavor" of the program without excessive tedious detail, some of these phases will be skipped over rather lightly; others will be described in some detail. Table 2 presented the lists representing behaviors to the P level which are constructed by C61. Reference to this table will aid the reader in following the discussion below.

In the initialization phase three routines are involved -Z0, C21 and D4. Z0 remembers the focus object and its designation as a member of the set of objects defined by the concept. C21 creates routine Z7 which establishes the initial conditions within the subject. D4 determines whether the subject is to continue on to phase two.

The processing of the focus-object information by Z0 is accomplished by four P level routines—P21, P61, C31, and P62. Because the subject and the experimenter both manipulate objects, it was necessary to design the program to separate information received from the external environment into its external representation and the subject's internal representation. Thus, P21 creates a copy of the focus object with its dimension values in dominant dimension order and also creates a dummy description list indicating that the object is a member of the set defined by the unknown classification rule. If one does not make this separation, descriptive information created by the subject becomes attached to the object in the external world, an undesirable situation. P21 leaves the name of the subject's representation of the focus object in M1-N and its set membership in M1-D. The memory process, P61, remembers the focus object under an attribute of the problem list, and C31 places the name of the focus object at the top of the memory entry point list. P62 then remembers the set membership of the focus object under an attribute of the focus object. At the present time, routine D4, inserted to keep the phase list structure consistent, is a dummy routine as the decisions the subjects make at this point have not been ascertained.

The procedure Z7 is created by the contexting routine C21; a detailed discussion of contexters is given at a later point in the present section. In Z7 the first routine is P191 whose inputs are the name of the memory entry point and K99, which is an input constant. The function of P191 is to create a working hypothesis from the focus object remembered by Z0. The working hypothesis is created by selecting the first K99 of the _m_ dimensions on the focus object and placing them on a separate list, the rationale being that some subjects deliberately work with less than the total number of dimensions and other subjects do so inadvertently. Hence K99 specifies how many of the possible dimensions are to be worked with throughout this attempt at attaining the concept. The working hypothesis is a list in its own right and a description of how it was created from the focus object; specifically the dimensions which have been removed in order to obtain the working hypothesis are made into a dummy description list. P191 leaves the name of the working hypothesis and the name of the description in working memory M1. Following P191 is a memory process routine, P63. P63 remembers the name of the working hypothesis under an attribute of the problem list devoted to the working hypothesis. At this point, the program needs to remember the description of the working hypothesis rather than something about the problem; hence P63 is followed by another memory process, C31, which will put the name of the working hypothesis into the memory entry point list, pushing down the list saving the name of the problem. C31 is then followed by memory process P64 which remembers how the working hypothesis was formed from the focus object itself. Thus, the problem is described by the working hypothesis, and the working hypothesis is described by how it was created. The rationale underlying this type of description is that other routines and contexting operations can utilize the information to determine what has happened and then can modify or create routines to change the behavior if necessary. It should be noted that the contents of working memory do not change during P63, C31, and P64; however, the memory entry point changes from the focus object to the working hypothesis to make it available to the next routine or the next phase.

The second phase is the object-selection phase which consists of three routines, Z1, Z2, and C32. Routine Z1 creates a search criterion from the working hypothesis; Z2 locates an object matching the search criterion; and C32 creates the routine D0 which determines whether the subject can proceed to the next phase. The procedure Z1 consists of four routines: P131, P141, P151, and P64. The inputs to P131 consist of M10, the memory entry point, and K98, a constant specifying the number of dimensions to vary. In the normal conservative-focusing strategy K98 would be one; however, it can be set to any number up to the number of dimensions on the working hypothesis. Note that K98 $\leq$ K99. M10 contains the name of the working hypothesis at the top of its list and it is from this hypothesis that P131 will select the dimensions to vary. Because P21 had arranged the working hypothesis in dominant dimension value order, P131 merely needs to select the first K98 of these dimensions, thus implementing the dominance feature in the program; i.e., the first dimension value on the list is the most dominant and the last is the least dominant. P131 creates a copy of the working hypothesis and puts the name of this copy into working memory on M1-N. It also creates a dummy description list, a DDL, on which it lists the names of the K98 dimensions which are to be varied. The DDL is only a partially completed list which eventually will become a "from-to list"; i.e., it will name the dimensions, tell what their values were originally, and tell to what value they were changed. However, P131 only places the names of the dimensions to be worked with on the DDL list. The output from P131 is left in M1-N and M1-D. Routine P141 takes the information from M1 and extracts the dummy description list containing the names of the dimensions to be varied. It then enters the dimension list, M13, which has been stored on the problem list L100 under the name of the external environment, ascertains a given dimension and the values available other than those of the focus object, and, if there are more than two values, selects a dimension value on the basis of its dominance value. The value found is then added on the DDL under the "changed to" attribute and the value on the focus object is stored under the "changed from" attribute. Upon the completion of P141, a copy of the working hypothesis is in M1-N, and the dummy description list is in M1-D. P151 is the routine which varies the dimension values to create a search criterion from the working hypothesis. It receives the working hypothesis and the DDL through M1 and uses the "from-to list" to change the dimension values on the copy of the working hypothesis to their new values, thus accomplishing the dimension variation. If an initial input flag dealing with awareness indicates that the subject is aware, the total DDL is placed in M1-D along with the name of the search criterion which has been created. It should be noted that the working hypothesis is not disturbed because the changing of dimension values occurs on a copy of a working hypothesis known as the search criterion which will be used to locate objects on the board. If the awareness flag indicates the subject is not aware, it is then assumed that he has inadvertently varied more than one dimension, even though he believes he is only varying a single dimension. It is very common in the protocols for subjects to choose objects which vary on more than one dimension, even though they believe they are searching for an object which varies on only one dimension. Possibly this is a perception problem; however, because we have eliminated perception, it is handled in this somewhat mechanical fashion. If the subject is unaware, routine P151 deletes from the dummy description list all of the dimensions and their values other than the first one. From this point on, the subject's description of what he has done indicates that only one dimension has been manipulated.

P151 is followed by a memory routine, C32, which places the name of the search criterion on the memory entry point list with a push down of previous information. C32 is followed by a memory process routine, P64, which describes the search criterion with the DDL in the working memory.

The remainder of the simulation program operates in a fashion quite similar to what has been described above; as information is created or received from the external world, it is initially left in the working memory, and the routines which process this information create a description and leave it in the working memory. Depending upon the information and its use, it either is left in working memory for subsequent routines to pick up and use as information, or, usually at the end of a series of routines, is attached to a previous unit of information through a memory process and the memory entry point list.

Phase three involves the use of an interesting contexter whose operations will be described in some detail. At the time phase three is entered, routine Z2 has located an object meeting the search criterion and has stored it in short-term memory under an "object found" attribute of the search criterion. The name of the object found has been stored in the memory entry point list, and the object has been described by the dimensions of the working hypothesis that were varied in order to find it.[1] The first routine in phase three is procedure Z3 which consists of processes P71, E93, and P62. Routine P71 is a memory output process which transfers the subject's name of an object to the output buffer E1 from which the experimenter will receive the information. Routine E93 is an experimenter routine which acquires the name of the object chosen from the E1 buffer and compares the dimension values of the object with those of the concept to ascertain whether or not the object contains the dimension values of the underlying classification rule. E93 creates a dummy description list similar to those used in the past which contains a designation attribute and a value of yes or no for the set membership of the object choice. E93 also returns the subject's name of the object to M1-N so that the subject may associate the designation with the object he has presented to the experimenter. Because the memory entry point contains the name of the object found, a memory process, namely P62, can be used to attach the experimenter's designation in working memory to the object choice in short-term memory.

[1] The reader should refer to Figure 3 to trace the levels of short-term memory involved.

The experimenter's designation of the object is information from the external world, hence it is mandatory that C61 insert a contexting program at this point. (Again some corners have been cut in that the appropriate contexting routine for this situation, C38, has been pre-programmed; whereas in a more sophisticated program the C61 contexter would analyze the total situation and create the contexting routine. That sophistication in program development has not been reached.) The contexting routine C38 will create the routine Z4 which is the reaction of the subject to the experimenter's designation of the object. Initially C38 creates a description of the characteristics of the required Z4 routine. C38 then receives from the input list its own location in the phase list that the interpreter is currently executing. Using this information, C38 ascertains whether or not the next symbol on the phase list has a description matching that of the routine Z4 which it wishes to execute. If the routine following the C38 contexter is to be a Z4 routine, it will be removed from the strategy and its symbol replaced by the symbol representing the new Z4 which will be created. If no Z4 symbol follows C38, as is the case the first time through the phase, the symbol for the Z4 routine is inserted on the phase list. Notice that at this point the phase list merely contains a symbol whose description indicates what the symbol should try to accomplish; however there is no executable sub-routine associated with the particular symbol. The phase list is also described through the use of the DDL technique to indicate that a routine has been either inserted or replaced on the phase list. The long-term goal is for contexters to utilize this change description to ascertain what has occurred during the execution of a program. The contexter C38 uses the memory entry point to obtain the name of the object and, through a descriptive attribute, ascertain whether it is a yes object or a no object. If the object was designated a no, C38 determines whether or not the subject was aware; and if the subject was aware, C38 checks to see whether the number of dimensions varied was equal to one or not. If it was greater than one, a situation exists from which no information has been gained, namely that the subject consciously varied more than one dimension and received a no. Therefore, he does not know which of the two dimensions is the relevant one. In this situation, C38 will pop the memory entry point back to the working hypothesis so that phase two can be executed again. If the subject is aware and varies only one dimension, or if the subject is unaware, the program returns to the creation of Z4. Again, difficulties have been circumvented by merely inserting routines that we know are necessary to accomplish the reaction to object designation. A routine called P502 is inserted to recall the concept designation. It is followed by C41 which pops up the memory entry point from the object found to be the working hypothesis. Popping the memory entry point is necessary because Z4 must have both the object designation and the working hypothesis in order to react to the object designation. The next routine inserted is P96 which uses the information about the working hypothesis and object designation to flag the dimension values involved as relevant or irrelevant. Following P96 is P91 which looks at all possible values of a dimension and checks whether they are marked relevant or irrelevant. If all values are marked, the dimension itself is then marked as relevant or irrelevant. However, if any dimension value is still untested, P91 will not attempt to mark the total dimension. We have found that many subjects will not consider a dimension to be relevant or irrelevant until they have checked all _n_ dimension values. If the subject is unaware of the number of dimensions actually varied, no further processes are required in Z4. However, if the subject is aware, he then also normally realizes that any dimension flagged irrelevant is no longer of concern in selecting objects and a routine called P100 which removes an irrelevant dimension from the working hypothesis is inserted after P91. The net effect of P100 is to enable the subject to choose objects in phase two which vary in two, three, or four dimensions from the focus object even though the subject is actually varying only one dimension; irrelevant dimensions no longer enter into any of his decisions. One can obtain what look like rather peculiar object choice sequences; however, the subject is truly varying only one dimension. If routine P100 has been inserted, it will be followed by a memory process, P64, which remembers the description of the dimensions removed from the working hypothesis so that at some later point a routine can put these dimensions back in again if necessary. The final operation performed by C38 is to put a terminal symbol on routine Z4 so that it can be properly terminated by the interpreter at execution time. At this point a rather tricky operation takes place; namely, C38 creates the next routine to be executed and when C38 terminates, the interpreter executes from the strategy list the routine which C38 has just created.

Phase three is terminated by procedure D1 which determines whether enough information is available to present a concept. D1 consists of a single decision process routine, Q101 which uses the memory entry point to obtain the name of the working hypothesis. Each dimension of the working hypothesis is checked to determine whether it has been flagged relevant or irrelevant. If all dimensions have been flagged, sufficient information is available to present a concept. Such a test is rather stringent as it requires the subject to vary all dimensions of the working hypothesis prior to forming a concept. Experienced human subjects do vary all dimensions as they know the concept must consist of the relevant dimensions. The result of D1 is an indication to the interpreter to either continue to phase four or to return to phase two and vary additional dimensions. Note again that a phase terminates in a decision routine.

There is actually little variation in the routine Z4 created by the contexter C38 during the first pass through the program. However, once phase four has placed previously unused dimensions on the working hypothesis, the Z4 routine can vary slightly depending upon the decision net through which it passes. In the future C38 will be made much more extensive; the rather rudimentary contexting operation used reflects current lack of understanding of the mechanisms involved in the subject's reaction to the experimenter's designation of an object. The only other contexter of any consequence in the program is C22 which creates process Z6, the reaction to a concept designation. C22 operates in much the same fashion as does C38, using various factors such as the number of dimensions varied, the number of untested dimensions, and the subject's awareness, and creates routine Z6.

Procedure Z6 is the subject's reaction to a concept which has been designated by the experimenter as incorrect. The procedure consists of P level routines Q41, P181, and C31. The contexter routine C38 has retained the memory entry point to the focus object so that Q41 may inspect it. Q41 uses the dimension values of the focus object to ascertain which dimensions have not been varied and creates a dummy description list containing their names. The number of untested dimensions to be used is controlled by the parameter K97 which specifies how many of the available dimensions to use. P181 uses the dummy description list created by Q41 to restore K97 dimension values to the working hypothesis. It partially undoes the work done by P191 in procedure Z7. If all dimensions have been varied and the concept is incorrect, Q41 assumes a dimension has been mis-flagged and P181 makes a copy of the focus object for the next working hypothesis. C31 places the name of the new working hypothesis in the memory entry point. P64 remembers how the new working hypothesis was created. The strategy now returns to the beginning of phase two, and the total process, less much of the previous contexting in C61, is repeated until the concept is attained.

The paragraphs above have presented the major features of the computer model; a description of the lower level programs which the computer program actually executes would be meaningless to readers without sophisticated knowledge of IPL-V and is beyond the scope of the present report.

SUMMARY OF CHAPTER

The computer model of concept-attainment embodied in Mark IV, Mod 2 consists of the breakdown of the behaviors representing a learning strategy, a hierarchy of contexting routines which supervise the behaviors necessary to attain the concept, and a model of the memory processes to implement the other two aspects of the model. Bruner's conservative-focusing strategy is represented by a strategy list containing symbols representing the fractionation of the behaviors involved in the concept-attainment process at the lowest level of fractionation. Routines represent tasks required by the information processing language rather than behaviors within the concept-attainment task and cannot be considered a part of the model itself. The behaviors in the fractionation of the conservative-focusing strategy essentially represent the operational or doing aspects of the concept-attainment process as these procedures and processes are primarily concerned with information processing of one type or another and are not concerned directly with goal attainment. One of the difficult tasks in the present project is recognizing that one does need to separate the operational information-processing tasks from those tasks associated with attaining the goal. The latter are embodied in the computer model as a hierarchy of contexting routines whose function is to select the appropriate behaviors, supervise their execution, and monitor the goal-directedness of the resulting behavior. Several levels of contexters are involved in the current model. The high level contexters essentially are designed to translate the experimenter's verbal instructions into a skeleton strategy for behavior. The second level contexters create routines associated with initializing a problem and analyzing the behaviors in a completed problem, and the third level contexters create situationally dependent procedures in an attempt to adapt behavior to the situation. Although the hierarchy of contexting routines is rather rudimentary at the current time, the distinction between operational aspects of learning and the contexting aspects of learning is held to be a very crucial distinction not previously made.

Because information, both acquired and internally created, plays such a crucial role in the concept-attainment process, it was necessary to create a model of memory which would enable the computer program to remember and to recall this information. The memory model created consisted of three levels: a working memory, which is a buffer-type memory; a short-term memory in which all of the information relative to a given concept-attainment task has been stored; and a long-term memory in which learning strategies and certain crucial pieces of information relating to them are stored for use in solving subsequent similar problems. In the present model, the working memory serves primarily as a holding or communication device for information which is to be passed from one behavior to another behavior within a section of the computer model. The short-term memory has been constructed in a circular form so that any point in the memory structure looks as if it were the beginning of an information storage tree. Several generalized memory processes have also been programmed to permit the model to remember and recall information within this circular memory structure. Although the model has not solved the problem of how a human enters memory under a given set of circumstances, it does include a memory entry point scheme for keeping track of subject's location within memory, once memory has been entered. The majority of work on a model of memory has been devoted to the working memory and the short-term memory, and the long-term memory has not been modeled in any great detail. The computer model, although somewhat rudimentary at the current time, has been designed with considerable flexibility so that it has capability for expansion without sacrifice of the capability already acquired.

The Mark IV, Mod 2 version of the concept-attainment model can reproduce a wide range of the behavior observed in the "think-aloud" protocols collected from human subjects. The range of behavior is accomplished with a relatively small number of computer routines, some of which, such as the memory processes, are completely general while others, such as contexters, are very specific. Unfortunately much of the variability is controlled by the three input constants and the awareness flag; however, even this is encouraging in that so much variability can be controlled by so few parameters. The long-term goal is to eliminate such parameters and utilize only generalized routines to accomplish what Newell [1965] has called a "solution by understanding."

III
A CHRONOLOGY OF THE DEVELOPMENT OF THE COMPUTER MODEL

The computer models constructed by the project staff have been designated by a Mark and Mod system for ease of reference. Each version of the model has been based upon a combination of data from protocols, analysis of previous programs, and clinical judgment. Because of the complexity in coordinating the various ideas involved in each computer program, extensive use was made of flow charts maintained in large work books. Most of the considerations entering into both the preliminary and final design of a particular Mark and Mod were recorded in the flow chart books. The specific details of each completed program have been recorded as line by line annotations in the program itself. In that the flow chart books represent a chronology of the ideas arising in the project, the present chapter has been organized by flow chart books such as CASE 1, CASE 2, where CASE is an acronym for concept-attainment simulation experiment, and by Mark and Mod corresponding to the flow chart books.

MARK I, MOD 0

The first program in this series, Mark I, Mod 0, was written by the author primarily as a device through which the programming language of IPL-V could be learned. The resulting program did not represent a very sophisticated level of programming in IPL-V but did serve as an entry point into the general area of simulation and computer modeling. The original program modeled an experimental situation which used a board consisting of sixteen objects, each possessing four binary dimensions. The program was not based upon any clear-cut theoretical point of view; instead the processes programmed were essentially what the author thought he would do in solving these kinds of problems. Many of the internal details were programmed merely to accomplish a given task without any consideration of modeling a psychological process. The resulting procedure was essentially a wholist strategy as described by Bruner et al. [1956].

In order to develop the Mark I, Mod 0 concept-attainment program, the following assumptions were made:

1. Characteristics of the concepts
   a. All concepts are conjunctive.
   b. Concepts involving only one or all four dimensions are not permitted.
2. Formation of concepts
   a. All tentative concepts are generated from dimension values of the focus object.
   b. Only information on the focus and yes cards are used to ascertain the defining concept.
3. Memory capabilities
   a. The subject has a perfect memory of all information obtained.
   b. The subject remembers all combinations of dimension values that have been used and does not repeat the use of a particular combination of dimension values.

The physical objects involved were represented as lists of symbols, each object becoming a list of symbols which represented dimension values. For example:

```
C6   0
     A_1
     A_2
     N_3
     A_4     0
```

where: A_1, A_2, A_4 are the first level of dimensions one, two and four; N_3 is the second level of dimension three. The board became list B1 of the symbols C1 through C16. A number of lists were maintained to store information regarding rejected concepts and object choices. These lists were Y0, the yes-object list, N0, the no-object list, and Q0, the rejected-concept list.

Analysis of information-processing aspects of the concept-attainment task indicated that a majority of the tasks could be accomplished by two utility sub-routines. The first, R1, could randomly replace n symbols on a list by a special symbol, B0, effectively removing a symbol from the list without modifying the length of the list. Creation of a tentative concept was accomplished by making a copy of the focus object and applying R1 to blank out several attribute values. The remaining dimension values then formed a tentative concept (T1). For example:

```
T1   0
     B0
     A2
     A3
     B0    0
```

The second routine, R2, compared a set of dimension values possessed by a symbol to those possessed by the symbols on a list representing the board. The output of R2 was a list of symbols whose dimensions had the same value as those specified by the input symbol. A typical function of R2 was to search the board for all objects having the same combination of dimension values as the tentative concept.

The information possessed by the experimenter consisted of only two lists, P4, the correct-concept list, and C0, the list of objects which were members of the set defined by the correct concept. The only functions performed by the experimenter were to inform the subject whether an object choice was a member of the set defined by the correct concept and to indicate the correctness of the concept. Thus, the "experimenter" could be represented by two applications of the R2 sub-routine. The "subject" was a routine which obtained tentative concepts from a sub-routine based upon R1, selected objects on the basis of the tentative concept, presented them to the experimenter, and ascertained common dimension values within the yes-object list by means of the R2 routine. In addition, the subject made certain decisions concerning the course of the experiment. An exposition of the flow chart, Figure 5, will illustrate the role of the assumptions and of the lists described above.

Starting from point A, the subject used R1 to generate either a two- or three-valued tentative concept, T0, from the focus object. The subject then compared T0 with those concepts previously rejected and stored on list Q0. If the dimension value combination of T0 appeared on Q0, the present T0 was discarded and a new tentative concept generated through the use of R2. If all possible combinations of two-valued concepts were listed on Q0, T0 was shifted to a three-valued concept; the converse also applied. The generation and screening process was continued until an unused combination of dimension values on the focus object resulted. The tentative concept, T0, was then compared to the objects on the yes list to determine whether a sufficient number of yes objects also possessed the dimension values of T0. If the number was insufficient, either the tentative concept was discarded and added to Q0, or the subject proceeded even though T0 was probably not a good tentative concept. Upon surviving the initial screening, the tentative concept was employed, via R1, to search the board, B1, for objects possessing the same dimension values; the output of R2 was a suitable-object list. The objects on the suitable-object list were then sequentially presented to the experimenter to determine whether they were members of the set defined by the correct concept. Preceding presentation, each object was compared to the N0 list so that a no object was never presented twice. When an object received a yes response, it was added to list Y0 and the Y0 list was searched to determine how many yes objects then had the same dimension-value combination as T0. If the number agreeing exceeded a predetermined value, the tentative concept was presented to the experimenter for designation. Acceptance indicated that the concept had been attained and the experiment terminated. Rejection caused the tentative concept to be added to list Q0, the suitable object list to be erased, and a return made to point A for another attempt. When an object presented to the experimenter received a no, it was added to the no-object list and the list was counted. If the number of no-objects exceeded the predetermined value, the tentative concept was added to Q0 and the program returned to point A. The final operation in the program was to print the sequence of objects as they were presented to the experimenter as well as the designation they received.

The basic computer program for Mark I, Mod 0 consisted of only two major routines, S1 which generated the tentative concept T0 and S0 which was all the rest of the computer program. The program itself consisted of several hundred lines of IPL-V written in what one would call straight-line coding with little or no sub-routining. The only sub-routines involved were R1, R2, and Z1, the last being a routine to find the nth symbol on a list because this version of the IPL-V did not possess the equivalent J200 instruction. Because of the

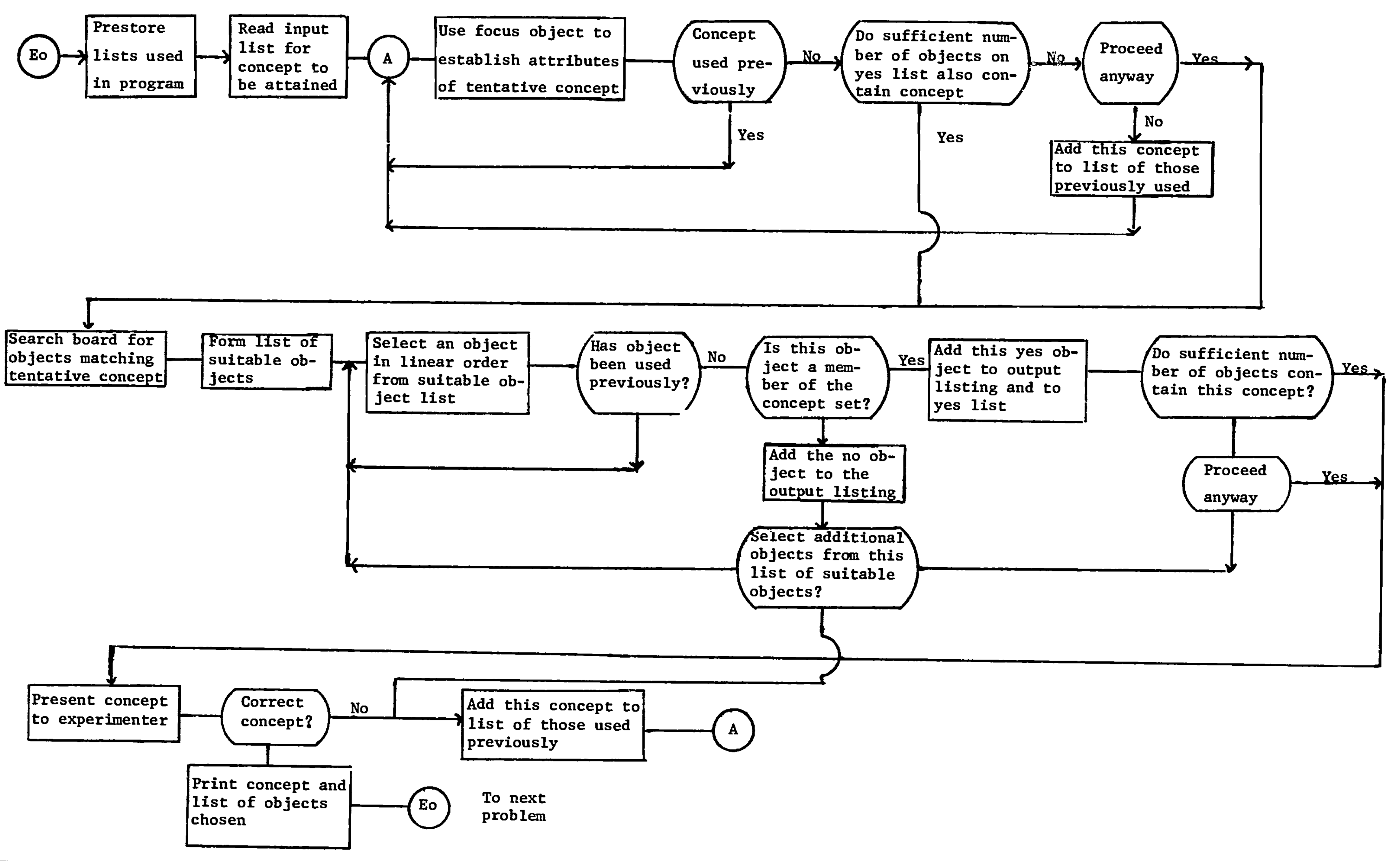
Eo
Prestore lists used in program
Read input list for concept to be attained
A
Use focus object to establish attributes of tentative concept
Concept used previously
No
Yes
Do sufficient number of objects on yes list also contain concept
No
Yes
Proceed anyway
Yes
No
Add this concept to list of those previously used
Search board for objects matching tentative concept
Form list of suitable objects
Select an object in linear order from suitable object list
Has object been used previously?
No
Is this object a member of the concept set?
Yes
Add this yes object to output listing and to yes list
Add the no object to the output listing
Do sufficient number of objects contain this concept?
Yes
Proceed anyway
Yes
Select additional objects from this list of suitable objects?
Present concept to experimenter
Correct concept?
No
Add this concept to list of those used previously
A
Print concept and list of objects chosen
Eo
To next problem

Fig. 5. Flow Chart IPL-V Concept Attainment Program Mark I, Mod 0

straight-line programming, the program was extremely clumsy to work with and minor changes necessitated a considerable amount of programming. The unsatisfactory nature of the programming aspects convinced us very quickly that one needed to sub-routine extensively in IPL-V, a fact well known to the Carnegie Tech group, but discovered anew in the present project. The use of random number generators to create a concept by eliminating one or more dimension values from a focus object also was not satisfactory. In addition, exhausting all possible combinations of two-valued concepts before proceeding to the three-valued, seemed excessively artificial. In order to prevent replication of object choices, the program used lists such as the suitable-object list, the yes-object list, the no-object list, and the rejected-concept list to act as a filter. Such a filter was merely a mechanical device to make the program work and had no clear psychological basis.

Despite all of its limitations, the Mark I, Mod 0 program was capable of obtaining concepts and, through the use of a large number of constants ("screwdriver" parameters") within the computer program, it could exhibit quite a variable sequence of object choices. However, this clearly illustrated that the overt behavior of a computer program could be quite misleading. The Mark I, Mod 0 program's overt behavior was quite similar to that of humans in terms of card choices but the internal mechanics, the sub-routining, and the psychological processes represented were clearly ad hoc.

The Mod 1 and Mod 2 versions of this program were aimed at exploring the capabilities of the program. Mod 1 permitted the computer program to generate the concept to be learned and create list C0 for use by the experimenter routine. The computer program could generate a problem and then go ahead and solve the concept-attainment task it had created. Because successive problem solutions were independent of one another, Mod 2 was designed to try to obtain some limiting distributions on object choices by allowing the computer program to repeat the same concept-attainment task a large number of times.

A fair number of computer runs were made using Mod 1 and 2 to ascertain the effects of the various parameters in the program. In that there was some concern about how much information people retain on a list, various lists were limited to lengths of four, five, six, and seven, and a number of studies conducted to determine whether the number of object choices necessary to attain a concept was a function of memory list length. Although it was possible to demonstrate statistically significant differences in number of object choices as a function of list length, it was felt that these studies fell into the broad, undesirable category of "screwdriver" research; this tack was very quickly dropped.

Many lessons were learned from the initial program, most of which were related to the necessity of studying the concept-attainment process and trying to break it down into reasonably sized units of behavior. The excessive length of the routines in the original version confounded many different aspects of the task itself. Therefore, in the next stage a process of fractionation was started which has not yet led to complete satisfaction with the level of partitioning with the program. The other major lesson learned was the unsatisfactory nature of ad hoc procedures for generating hypotheses. The use of random number generators to eliminate particular dimension values of the focus object did not seem particularly sophisticated psychologically and could not be justified very long as a model of human behavior.

The general dissatisfaction with Mark I led the author to begin a long-term project with the eventual goal of developing a reasonable computer model of concept attainment. The remainder of the present chapter will be devoted to a chronological description of the various Marks and Mods through which the program has developed as well as some of the preliminary thinking which went into the development of each version of the program.

CASE I, DATED 10 AUGUST 1964

In this book a hierarchy of behavior within the concept-attainment task was developed. The concept-attainment task was represented by learning strategy S which consisted of procedures at the Z level, processes at the P level, and small information processing routines at the R level. In this version of the strategy list, the sub-routines themselves were directly executable as IPL-V routines, and each sub-routine assumed it knew where necessary information for its task could be located and where all information was to be stored. A strategy list was represented symbolically as follows:

S1	Z1	Form a current hypothesis.
	9-10	Check to see whether all possible combinations have been exhausted.

Z2 Search the board for objects matching the search criterion; in other words, create an available object list.

9-11 Present the object to the experimenter for designation.

Z3 Use the designation information to create an intersection of the yes objects and the focus object.

9-12 Decide whether to present the concept. (If a sufficient number of yes objects agreed with the concept, it would be presented.)

9-13 Present the concept to the experimenter for his designation.

Z4 On the basis of the correctness of the concept, decide what to do. (If the concept was correct, one merely exited from the program; if it was incorrect, one checked to ascertain whether the two-valued combinations had been exhausted; if they had, the number of dimensions in a concept were changed from two to three, or vice versa.)

Under this scheme the routines represented by local symbols (9-10, etc.) were "housekeeping routines" whereas the Z's were psychological procedures. Each of the procedure level routines, namely the Z's, were broken down to a number of sub-routines at the P or process level. A few of these P level subroutines will be described to indicate what was involved. Procedure Z1 consisted of routines P100, P101, P102, and P103. P100 used the focus object to generate a tentative hypothesis. P101 checked whether or not the hypothesis had been previously rejected. P102 checked whether the tentative concept agreed with a sufficient number of yes objects. P103 determined whether the dimensions blanked out on the tentative hypothesis had been blanked out over all possible combinations of values in the past. If so, it returned to P100 to create a hypothesis not involving that particular dimension. Z2 consisted of routines P108 which created a list of objects matching the current hypothesis and P109 which located an object choice not having a specific value. In other words, P109 obtained an object from a list of those matching the current hypothesis, checked to see that it had not been previously rejected, added it to the suitable-object list, and created a count of the number of objects available from this list. The procedure Z3 determined the course of action as the function of the designation of an object's set membership and consisted of P105 and P106. The former added the object choice to the object-presented list. The latter processed an object choice designated by the experimenter as not being a member of the set defined by the concept. If an object choice received a no, several options existed, one of which was to present the next object on the suitable object list to the experimenter. If that list was exhausted, the program returned to Z1 and created a new hypothesis. In procedure Z4, which determined the course of action based upon the designation of the current concept, routine 9-13 presented the concept to the experimenter for designation and routine P107 added a rejected concept to a list containing rejected hypotheses.

It should be noted that some of the inputs to the P level routines consisted of long lists of separate cells containing data, such as M15 for the current object and M25 for the list of objects defined by the current concept. Such a scheme was an initial attempt to avoid having the programs contain all of the locations they needed for execution.

Essentially what had been done in CASE 1 was repackaging the original Mark I, Mod 0 program by fractionating the very long program lists in the earlier version. It still used the wholist approach in which many lists were maintained and the hypothesis generation procedures were random number generators. The major idea to come out of this edition of the program was that of the strategy list in which the major procedures were represented by Z level routines, the processes were represented by P level routines, and information processes were represented by the R level routines. The Z level routines essentially set up the inputs to each one of the P's prior to execution of the P.

The cleaned-up version of Mark I, Mod 0 was never formally designated as a particular program in the sequence as it served as a tool through which Mr. Martin, the programmer, learned IPL-V.

CASE 2, DATED 9 NOVEMBER 1964

The Mark II version was fully conceptualized and flow charted but never programmed because of certain limitations which became apparent in protocol analysis. The original idea underlying Mark II was to incorporate into the program a rather detailed strategy for searching the board in the external environment. It was thought that there was a considerable dependence of the problem solution upon the method by which the subjects searched the board. The original Mark I version did a linear search of the board and the solution of the problem was quite dependent upon the initial object choices. Experiments were run in which subjects' eye movements were observed and subjects' object choices were analyzed to ascertain whether or not a particular search pattern was used. Analysis of the data revealed that the sear~ strategy was not a very critical part of the overall concept attainment process. Subjects using a wide variety of scanning procedures made essentially the same card choices. Therefore this aspect of the program, though flow charted, was not implemented. Other outcomes of Mark II were much more important in terms of the long-term development of the computer model.

The strategy list idea was much better developed in Mark II than in the previous edition. The S, Z, P, and R breakdown was maintained, but the inputs of the P's, which were formerly programmed in IPL-V within each of the Z's, had been extracted from the Z list so that the strategy list now consisted only of symbols representing routines at the various levels in the strategy. The strategy list was now executed by an interpreter written in IPL-V [Baker and Martin, 1965] which took over the functions of input-output previously embodied in the Z routines themselves. The interpreter developed at this point has been used throughout the rest of the computer modeling project with very few changes. The basic mechanism of the interpreter was to ascertain whether or not a symbol was described. If the symbol was described, it then meant that this symbol was merely a holding list for lower level routines. Therefore, the interpreter would work its way down through a hierarchy of symbols until it found a routine which was non-described, in other words a symbol in the list which was an executable IPL-V instruction. The pseudocode system described earlier was developed at this point as was the description list containing A1 with its value list for inputs and A2 with its value list for outputs. Attribute A3 of the description list was designated to describe the characteristics of the routine, although at this time no attempt was made to describe the routine itself.

The three level model of memory was the second major development in the Mark II version of the program. The description of the working memory (M1) indicated that it contained only what the subject "sees" or "hears" from the external world and that sequential sub-routines within a procedure used the working memory to hold information. Another basic idea was that information passing from working memory to short-term memory had to be recoded in some sense. The initial basis for deciding whether or not to recode information was to apply utility values to the information. If the information had sufficient utility value to the "subject" it would be recoded and stored away. The utility-value idea was not explored further as other pressing problems forced it aside. At this point the short-term memory merely consisted of simple lists such as an object-choice list, a hypothesis-generation list, and a concept list. The lists themselves were described in terms of their length, type of information contained on them, etc. The third level of memory, long-term memory, was slighted because the model did not yet require it. It was also visualized that some general-purpose memory routines would be required to search each level of memory, the thought being that separate memory processes would deal with working memory, short-term memory, and long-term memory. The basic memory structure consisted of a list called M0, which represented memory, and on the body of this list were three symbols: M1 for working memory, M2 for short-term memory, and M3 for long-term memory. A number of different memory structures for short-term memory were compared in order to ascertain their programmability, but the simple list structure was finally selected in which each symbol on the body of a list represented a particular piece of information.

Because Mark II used a strategy in conjunction with the special purpose interpreter, it was no longer possible to execute the program directly as was the case in Mark I; hence, it was necessary to establish an executive program for the concept-simulation program itself. The executive for Mark II consisted of routines which would erase the bodies of the various lists involved such as the problem-specification list M10, the hypothesis list M11, and the object-chosen list M12. A second section read in the problem specifications such as the focus object and the correct concept. The final task of the overall executive was to give the strategy list to the interpreter for execution. After execution of the strategy list, the overall executive printed the various lists involved in order to present the course of action within the program. The executive was an important step as it separated the housekeeping activities associated with running the program from the actual execution of the strategy list.

The breakdown of the concept-attainment

Table 4

Symbolic Representation of Strategy Used in Mark II

S1	0	
9-1	21	Create a tentative hypothesis.
9-2	22	Select an object using the hypothesis.
	D1	Determine whether the object can be used.
	9-2	No.
	Z3	Yes, have experimenter designate the object.
	D2	Determine course of action.
	9-3	No.
	D4	Present the concept for test.
	9-1	No.
	0	Yes.
9-3	D2	Use the correct hypothesis.
	9-1	No.
	9-2	Yes.
	0	0

process, originally started in the CASE 1 book, was further extended in the Mark II edition. The strategy in Mark II was represented by list S1 and is presented in Table 4. A significant change in the mechanics of the strategy list from Mark I can be seen in Table 4. The symbols such as Z1 and D1 represented procedures, but the local symbols 9-1 and 9-2 no longer represented "housekeeping" routines. The local symbols now represented links in the strategy list which permitted procedural loops. A rule was established that decision procedures, D routines, must be followed by local symbols indicating where the program was to go if the decision resulted in a no. If the result was a yes, the local symbol was ignored and the next procedure in the strategy was executed. The major procedures in Mark II were broken down as follows: Z1 consisted of four routines. P1 created an _n_ valued hypothesis, again using a random number generator similar to that of Mark I. Q1 determined whether the hypothesis had already been rejected in the past. Q2 ascertained whether either all the three-valued or all the two-valued hypotheses had been attempted and the dimensionality of the hypothesis needed to be changed. A3 tested whether a sufficient number of yes objects contained the dimension values of the tentative hypothesis. The procedure Z2 consisted of processes P2, P1000, Q4, and Q5. P2 established a search criterion and created an XY origin for the search routine P1000 to search the environment in some XY pattern; Q4 determined whether or not the dimension values of the object found contained the dimension values of the hypothesis; Q5 occurred only if objects matching the hypothesis were not found and prevented a loop from developing when searching on the board. Procedure Z3 consisted of routines P9 which had the experimenter designate the object, P3 which described the object found as a yes object or a no object, and P6 which counted the objects found for each hypothesis and indicated how many were designated by a yes and how many were designated by a no. Procedure D1 contained Q6 which merely ascertained whether the object presented received a no designation and routine P4 which removed the object from the object-presented list if it had previously been designated as a no object. Procedure D2 consisted of routine Q7 which determined whether or not the object had been designated as a yes or a no by the experimenter. If the designation was a yes, the program proceeded further; if not, it returned to the hypothesis-generation routine. Procedure D3 consisted of Q8 which checked to determine whether or not a given hypothesis had yielded too many no objects and, if so, dropped the hypothesis from consideration as a concept, thus preventing excessive searching with hypotheses which would not yield useful information. Routine P5 was a bookkeeping-type routine which placed the focus object at the top of the arbitrary XY origin list and enabled the hypothesis - generation routine to start again. Procedure D4 contained P6 which presented the concepts to the experimenter, P3 which stored the experimenter's designation of the concept in short-term memory, and P5 (the same P5 as in D3) which was a housekeeping routine for setting up the lists in working memory for the generation of a new hypothesis.

It can be seen from the breakdown of the program that the original Mark I program was fractionated in order to reduce the large computer program into more manageable sections of programming. Although the redesigning and re—sub-routining of the Mark I program into the Mark II version resulted in the strategy list and the interpreter system, the basic design of the concept-attainment model remained inadequate. The mechanics through which hypotheses were generated and the method for retaining information and utilizing information clearly were not going to be adequate over a long period of time. It was clear that a major restructuring of the total computer program was necessary. Therefore, although Mark II was flow charted—and many new ideas were gained from the flow charting—it was never written into IPL-V code to be run on the computer. An additional reason for not programming the Mark II version was that analysis of the protocols that had been gathered by Mr. Pratt indicated most subjects were using the conservative-focusing strategy. Therefore, a decision was made to abandon the earlier "Baker's own version" and standardize upon the conservative-focusing strategy for the next major development.

Although Mark II was abandoned, it nonetheless produced three major conceptual changes upon which further progress depended. First, the three-level model of memory consisting of working memory, short-term memory, and long-term memory was conceived. Second, short-term memory consisted of a number of simple lists whose descriptions contained information to be used by a number of subroutines. Third, the special purpose interpreter was developed which permitted one to represent behavior as symbols on a strategy list and execute the strategy in a sophisticated fashion. The mechanical details of the memory structure were radically altered in later versions of the program, but the interpreter has remained virtually untouched throughout the project.

CASE MARK III, MOD 0, DATED 3 FEBRUARY 1965

The Mark III, Mod 0 version of the concept-attainment program was the first to model the conservative-focusing strategy and was a major repackaging of the total program. The protocols were studied to determine the characteristics of the conservative-focusing strategy. The "normative" conservative-focusing strategy was then flow charted, and the flow charts broken down to the P, Q, and R levels in order to ascertain the basic sub-routines necessary to perform the conservative-focusing strategy. A serious attempt was made to match the procedures at the Z level and the processes at the P level to behavior observed in the protocols collected in the previous several months. Within the P level routines, a completely new set of underlying information processing routines, the R level routines, were generated such as remembering, recalling, data transmission routines, comparison routines. In addition to this fractionation, even further fractionation was made in that lines of computer programming appearing in more than one R routine were extracted and given the name of G routines so that information processing routines could be constructed by assembling several G's. The scope of this restructuring of the program may be illustrated by the observation that only one R level routine from Mark I was retained for use in Mark III, and it was highly modified.

The symbolization of the board containing the objects was restructured, as the previous system of using the letter A to represent one value of a dimension and the letter N to represent the other value was not very satisfactory and did not allow for more than two dimension values. The previous scheme was replaced by a hierarchy of symbols representing the structure of the dimensions and their values. In the new scheme the symbols E10, E20, E30, E40, and E50 represented the dimensions themselves such as color, position, and number, and under each of these was a further breakdown. For example, E11, E12, and E13 represented the specific values of the color dimension E10, yellow, brown, and blue. An object on the board was represented by O12, and on O12, the dimension values of the object were represented by the symbols E11, E21, E33, E41, and E52.

Table 5

Symbolic Representation of the Conservative-Focusing Strategy in Mark III, Mod O

S2	9-0	Conservative-focusing strategy		
	Z0	9-3	Problem specification	
		E92		Experimenter presents focus object
		P21	0	Remember focus object
9-1	Z1	9-3	Create object selection criterion	
		P31		Vary a dimension
		P41	0	Make copy of selection criterion
	Z2	9-3	Select an object	
		P51		Locate object meeting selection criterion
		P71		Remember object chosen
		P61	0	Describe object found
	Z3	9-3	Have object designated by experimenter	
		P72		Place on output channel
		E92	0	Experimenter designates set membership
	Z4	9-3	React to object designation	
		P81		Describe object's set membership
		Q11		Evaluate set membership
		9-1		
		P101		If yes, flag as irrelevant and remove value
		0	9-4	
	9-4	P91		If no, flag as relevant and revert value
	D1	9-3	On concept to be presented	
		D11	0	Check whether all dimensions used
	9-1		No, create new search criterion	
	Z5	9-3	Yes, determine whether concept correct	
		P121		Form a concept
		P73		Place on output channel
		E94	0	Experimenter designates concept
	D2	9-3	React to concept designation	
		P82		Remember designation of concept
		Q12	0	Evaluate concept designation
	0	0		

The working memory was redesigned at this point so that M1 listed only two elements. The first element was the name of the object, and the second element was a dummy description list containing holding symbols, for example 3671, for description lists of specific attributes and their values.

```
3671   9-2      0
 9-2    0
       A12
       V12     E11     0
       A13
       V13     E12     0
```

It was also decided at this point that all external information entering the system would enter through the working memory.

It should be noted that throughout Mark III there was no transfer of information between two Z level routines via the working memory. All information in working memory had to be remembered in short-term memory before the next Z routine could be executed. This was necessary because, if one was eventually to be able to interchange Z level routines, unstored data would make interchangeability impossible. Although the symbols representing major procedures were relatively consistent from program-to-program, the actual programming underlying Mark III, Mod O was completely different from that of the previous Marks and Mods, hence the reader should not infer the routines were the same. Certain labels were standardized in order to simplify our thinking about the problem. In the paragraphs below, each of the major procedures is described to indicate the sub-routines involved and give some indication of the design of the concept-attainment program at this point. The structure of the Mark III, Mod 0 revision can be seen in the strategy list S2 which is presented to the P level in Table 5.

The problem specification procedure Z0 consisted of routine E92 which placed the name of the focus object in working memory with a yes designation to indicate that it was a member of the set. P21 remembered the focus card as an object by placing it on the object-presented list and also remembered it as a hypothesis by placing it on the hypothesis list.

During a discussion of the concept-attainment program one of the students in the author's computer course mentioned that much of the behavior observed in the protocols appeared to be due to the people's using dominant dimensions; i.e., subjects have a hierarchy in which they chose the dimensions to be varied. For example, some subjects would vary color, then shape, then position, where others might vary number, then shape, then color, etc. [See Brian & Goodenough, 1929.] The suggestion appeared to be an extremely interesting idea, and a detailed analysis of the protocols indicated that it was an easily observed phenomenon within our data. Therefore, a decision was made to include the dominance factor in the system. It was a rather easy thing to incorporate from an IPL-V point of view because the J16 instruction selected attributes on a probabilistic basis. Each dimension and each dimension value was assigned a probability, and by using that probability in selection of the dimension and its value one implemented the dominance feature observed in the protocols. Therefore, the M13 list which contained the structure of the external environment was modified to include the dominant dimension attribute values required by the probabilistic instruction J16.

The search criterion based upon this technique was created by routine Z2 consisting of processes P31 and P41. P31 selected a dominant dimension value from the dimension values of the focus object and was set up to vary only one dimension, assuming that all dimensions were binary valued. The search criterion created by P31 was placed on top of the hypothesis list called M11. Some mechanical problems were encountered and it was necessary to place an additional copy of the search criterion, via P41, on the hypothesis list. The rationale was that the next execution of P31 would manipulate the dimension values of the copy at the top of the M11 list.

Another mechanical device used to prevent the program from testing the same dimensions over again was to set the attribute values for the dimensions used to a negative number in order to suppress their selection by the J16 instruction in IPL-V. A great deal of internal bookkeeping was generated in the program to restore positive values to the attributes so that the probabilistic mechanism could be used again by P31. Due to the lack of alternatives, it appeared to be a very useful way of achieving the dominance values.

The procedure Z2 was the object-selection part of the concept-attainment task and consisted of routines P51, P71, and P61. The process P51 used the search criterion generated by Z1 to search the board for an object having the dimension values given on the search criterion. A straight-forward linear search was used starting from the top of the list of symbols representing the board, and the first object matching the search criterion was the desired one. Because each search criterion differed from the previous one, there was little possibility of duplicating card choices or getting into various types of loops. The output from routine P51 was the name of the object, such as O16 or O21, and a dummy description list containing a copy of the search criterion to indicate how the object was chosen. Following P51 was a memory process routine, P71, which placed the name of the object found at the top of the object-found list, M12, pushing down the remainder of the list. The last routine in Z2 was P61 which attached the dummy description list to the object found under a search criterion attribute; i.e., P61 described the object by the search criterion that was used to find it. It should be noted in conjunction with routine P51 that an underlying IPL-V routine, called U20, was created to construct the dummy description list (DDL) which was stored in M1 and subsequently used by P61 for descriptive purposes. The rationale underlying U20 was that the creation of the DDL amounted to what Miller et al. [1960] had called "chunking." The name of the DDL represented a certain amount of information, and one dealt with the name of the DDL rather than with each individual item of information on its list.

Z3 was the procedure by which the experimenter designated the set membership of the object chosen by the subject. Within this procedure, P72 placed the name of the object chosen by the subject on the external communication buffer E1. Routine E93 matched the dimension values of the object the experimenter found named in list E1 with that of the correct concept. If the object contained all of the dimension values of the concept, it was then given a yes designation and the routine E93 placed the name of the object in working memory followed by a dummy description list containing the designation of yes. If the object did not contain all of the dimension values of the concept, the designation was a no.

The procedure Z4 was the subject's reaction to the designation of the set membership of the object choice. The first P routine within this procedure was P81 which remembered the designation of the object by the experimenter. P81 extracted the yes designation from the dummy description list in working memory and remembered it under an experimenter designation attribute of the object stored on the top of list M12. Routine Q11 performed the reverse process, recalling the experimenter designation attribute of the object and indicating whether its value was a yes or a no. The sequence of events seemed somewhat clumsy; however, P81 remembered the designation for future use and Q11 recalled it from short-term memory for immediate use. Q11 was followed by routine P101 which processed the yes designation of an object. If the object chosen was a yes, the subject knew the dimension value changed from the focus card was irrelevant and the dimension was removed from the copy of the focus card on the hypothesis list M11; also the dimension was flagged as being irrelevant. If the object chosen was designated a no, routine P91 was used rather than P101, and the dimension value of the search criterion reverted to its original value and was flagged as relevant. In either case the dominant dimension attribute value was set to the negative of its value to indicate that it had been varied.

Procedure D1 consisted of a single subprocess, D11, which determined whether or not a concept could be presented. The process merely checked whether or not dominant dimension attribute values of all dimensions had been set to negative. If they had all been set negative, this meant that all dimensions had been varied and it was possible to present a concept.

The next task was to create a concept and present it to the experimenter to determine whether it was the correct concept. The procedure Z5 accomplished this task, and the initial routine P121 in Z5 created the concept to be presented. The concept was created by searching list M13 (the description of the dimension value structure) for dimension values which had been described as relevant. A list was constructed and all relevant dimension values were placed on this list. The following process, P73, was a transmissive routine which placed the name of the concept on the output list E1 so that the experimenter could acquire it. E94 was much the same as E93 in that it compared the subject's concept and the experimenter's concept and, if the two were identical, placed a yes designation in M1.

Upon designation of the concept by the experimenter, procedure D2 was entered for the subject's determination of the concept designation. The first routine, P82, remembered the designation of the concept in short-term memory. The routine following it, Q12, recalled the designation and indicated to the subject whether it was a yes or a no. Because the strategy described above was a perfect conservative-focusing strategy, the program terminated after procedure D2.

It should be noted that in Mark III, Mod 0, even though the pseudo-code system with the interpreter was used, a significant number of routines possessed a knowledge of where various information was stored in short-term memory and where information should be left in short-term memory after computing. This was a carryover from Mark I which had proven to be very difficult to eliminate.

Several notes in the CASE Mark III, Mod 0 flow chart books discussed the idea of a contexting routine. The context routine was discussed as a routine which permitted a P level process to appear at several points in the program differing only in terms of input-output information given to the P routine. Such a consideration was generated by the observation that each of the memory processes had been programmed independently so that a routine storing information in Z1 might not necessarily be the same as a routine storing information in Z6. However, upon programming Mark III, Mod 0 and going through the actual IPL-V code, it was observed that the coding within the storage routines was essentially the same, varying only in the names of the lists used. If a context routine were available, it could set up the inputs to a generalized P routine. With this idea in mind, an attack upon the completed Mark III, Mod 0 was made to see what communalities existed throughout the computer program. It was quickly observed that the remembering routines of the original flow charts, called P61, P81, and P82, were essentially the same except for their inputs and their storage locations. A generalized memory storage routine, P60, was written to attach descriptions (DDL's) to information in STM. Plans for a separate context routine to set up the inputs on its I-0 list were flow charted. Due to the lack of an appreciation of the significance of contexters, the contexters to accompany the general purpose remembering routine, P60, were not programmed. Rather, the human programmer merely put the proper information on the I-0 list in order for P60 to run as if a contexter had prestored it.

The success with the generalized memory routine P60 prompted a study of the rest of the program to determine whether further fractionation and generalization at the P-Q level was possible. Routine P31 which generated a search criterion was fractionated into three programs called P131 which selected a dimension, P141 which used the dimension to select a dimension value, and P151 which used the description of what was to be changed to actually change a copy of the focus card and created a search criterion. It was also observed that attaching the search criterion to the description of the object was not particularly relevant as subjects in the protocols indicated that when they looked for an object the crucial information was the dimension that was being varied. Therefore, the idea of attaching search criteria was dropped; the dummy description list containing the dimension varied and the "from-to" information was attached instead.

As one can see, the Mark III, Mod 0 program was a significant change from the previous Mark I and Mark II versions of the program, especially in terms of the packaging of the program and the use of the conservative-focusing strategy.

Within the program a number of mechanical difficulties existed and the scope of many P level routines was too extensive. The mechanics by which the probabilistic selection process was performed and the internal bookkeeping necessary for the program to maintain a proper set of dominant dimension values were quite difficult to use. The use of the pseudo-code input-output lists was not consistent as a number of P's obtained information directly from short-term memory. The analysis of the completed Mark III, Mod 0 program showed that further fractionation of the P level routines was not only necessary but feasible. The execution of a strategy by an interpreter was shown to be a significant programming technique and was a major accomplishment of this version of the program. The insights gained in Mark III, Mod 0 were used to design the next computer model in the series, Mark III, Mod 1.

CASE MARK III, MOD 1, DATED 9 MARCH 1965

The features of Mark III, Mod 1 were as follows:

1. All information came into the program through the input list under A1; therefore, within a given P or Q routine, no reference was made to a memory list. This finally solved the problem of having some information built into the program and other information given to the program by the interpreter.

2. A maximum number of P and Q level routines left their outputs in the working memory, and storage in short-term memory was accomplished by a general purpose storage routine.

3. In the Mark III, Mod 0 routine the M13 list was an exception to the modular memory structure and had given us quite some difficulty in programming. In Mark III, Mod 1, the M13 list, with its dominant dimension attribute values, was forced into the modular structure.

4. The working memory became a standard communication device between successive P-Q level routines within a given procedure.

In Mark III, Mod 1 three basic routines were developed for use in a number of different situations: P60 remembered the description of a piece of information; P70 transferred information from one list to another with push-down of the recipient list; Q10 tested the value of a specific attribute on a description list in short-term memory; i.e., if one wanted to find the value of the experimenter designation attribute for a particular object, one could give the name of the object and the name of the attribute and determine whether it was a yes or a no. Because of the new basic routines involving transfer of information, a complete repackaging of Mark III was undertaken which was as extensive as that between Mark II and Mark III. Each of the P level routines on the strategy list representing Mark III, Mod 1 as given in Table 6 are discussed below.

The first process within Z0, the problem specification procedure, was routine E92 in which the experimenter presented the focus object to the subject and designated it as a member of the set. Both the name of the focus object and its designation were left in M1 by routine E92. A rather peculiar method for handling certain descriptive information was used in Mark III, Mod 1; at this point P63 would attach the DDL containing the designation of the set membership of the focus object to the focus object while it was in M1. The name of the focus object now with the experimenter desig-

Table 6

Symbolic Representation of the Conservative-Focusing Strategy to the P/Q Level for Mark III, Mod 1

S2	9-0	Conservative focusing strategy	
	Z0	9-4	Problem specification
		E92	Present focus object
		P63	Attach yes designation to focus object
		P71	Remember on M12 list
		P21	Copy focus object
		U11	Erase focus object description on copy
		P75	Remember copy on M11 list
		P22	Copy focus object
		P75	0 Remember copy on M11 list
9-2	Z1	9-4	Create object selection criterion
		P131	Select a dimension to vary
		P141	Select a new dimension value
		P151	Change value on search criterion
		P62	Attach how formed description
		P75	0 Remember search criterion on M11 list
	Z2	9-4	Select an object
		P51	Find object meeting search criterion
		P71	Remember object name on M12
		P61	Attach search criterion to object
		P22	Copy search criterion
		P75	0 Remember search criterion on M11 list
	Z3	9-4	Have object designated by experimenter
		P72	Place object name in output buffer
		E93	0 Experimenter designates object set membership
	Z4	9-4	React to object designation
		P63	Attach designation to object
		Q11	Recall designation
		9-5	Designated as a no
		P91	Mark dimension value as irrelevant
		P101	0 Remove dimension from search criterion
	9-5	P92	Mark dimension as relevant
		P171	0 Revert search criterion to original dimension value
	D1	9-4	Can a concept be presented ?
		D11	0 Determine whether all dimensions varied
	9-1		No
	9-2		Yes
9-1	Z5	9-4	Present concept to experimenter
		P121	Construct the concept
		P73	Place concept to output buffer
		E94	0 Experimenter designation of concept
	D2	9-4	Determine correctness of concept
		P63	Attach designation to concept
		Q11	0 Recall concept designation
	9-2		No
		X11	0 Yes, print history

nation attached to it was left in M1, and routine P71 placed the name of the focus object at the top of the objects-presented list, M12. Routine P21 then created a copy of the focus object whose description as a member of the set defined by the concept was erased by a utility routine called U11. Routine P75 placed the copy of the focus object at the top of the hypothesis list M11 where it served as the initial hypothesis from which dimension values were varied for creating search criteria. Because of some idiosyncratic programming later in the program, P75 was followed by a P22 routine which created another copy of the focus object and a P75 which also placed this copy at the top of the hypothesis list M11. The double copy was necessary in order to create a history of all search criteria used.

Procedure Z1 formed an object-selection criterion by varying a dimension of the hypothesis appearing at the top of the M11 list. Routine P131 selected a dimension to vary through the probabilistic selection device mentioned earlier. P141 used that dimension to select the value to which the hypothesis should be changed; for example, if the focus card was green, P141 selected red. Routine P151 did the actual changing of the working hypothesis by changing the "from" dimension value on the DDL given to it by P141 to the new value given by P141 on the "to" portion of the DDL. After P151 the contents of working memory were the name of the new search criterion and a dummy description list containing the description of how this particular search criterion was formed, namely the dimension varied, the original value (the "from"), and the new value (the "to"). P62 then attached this description to the working hypothesis while it was in working memory and P75 stored the search criterion at the top of the hypothesis list M11.

Z2 was the object selection procedure in which routine P51 linearly searched the board for an object which contained the same dimension values as the search criterion. It should be noted that the search criterion included all the dimensions of the focus object with one value changed. Routine P71 then placed the name of the object found on the object-presented list, M12. P61 then attached the search criterion to the object under a search-criterion attribute. P22 created a copy of the search criterion and its description and left it in working memory. P75 then placed this copy at the top of the working hypothesis list, M11.

Z3 was the procedure for experimenter designation of the object chosen. P72 took the name of the object appearing at the top of the object-presented list and placed it on the external communication buffer E1. E93 then compared the dimension values of the object presented with those of the correct concept and, if the concept was contained within the object choice, the object was designated as a yes, if not, as a no. The name of the object and the experimenter designation contained in a DDL were then placed in working memory.

Z4 was the reaction of a subject to the experimenter's designation of the object presented. P63 attached the experimenter's designation of the set membership of the object to the object as it remained in working memory. This mechanically turned out to be the same as attaching to the object itself because the symbol representing the object was the same in M12 as it was in M1. Having P63 working in M1 equivalent to its working in M12 was convenient from a programming point of view, but nonetheless was an idiosyncratic way of doing things. Q11 checked the value of the experimenter designation attribute of the object at the top of the objects presented list, M12, and ascertained whether it was designated yes or no. If the object was a no, the program jumped to branch 9-5 where routine P92 marked the dimension and the dimension value as relevant on list M13. Following P92 was routine P171 which used the information stored by P151 describing what dimension values were changed and reverted the value of the dimension previously varied to its original value. The hypothesis at the head of list M11 now had the same value as it had on the focus object before any changes were made. If the object was designated as a yes, the dimension varied was irrelevant, and P91 marked the dimension and its value as irrelevant. Following P91 in this leg of the decision point was routine P101 which removed the dimension from the working hypothesis at the head of list M11. Because the subject no longer needed to worry about the values of an irrelevant dimension in his object selection, it was removed from further consideration through this device. It should be noted that P91 and P92 were pseudo-codes for a general routine P90, differing only in the flag used.

After each object had been presented and the dimension values flagged appropriately, the subject checked to determine whether or not a concept could be presented; this was done through procedure D1. Process D11 merely checked whether all the dominant dimension attribute values had been set negative by the routines which created the search criteria. If all values were negative, all dimensions on the focus object had been varied and a concept could be presented. If not, the computer program would return to Z1 and choose another object.

If a concept could be presented, procedure Z5 was utilized. P121 obtained the hypothesis at the top of the M11 list, generated its dimension values, and used them as input to an R level routine which used each value in conjunction with the M13 list to determine whether the dimension value was relevant or not. If a dimension value was relevant, it was placed upon the concept list and retained as part of the underlying concept. The routine P121 generated all dimension values from the hypothesis list, and the net effect was a list of relevant values. Upon completion of P121, routine P73 transferred the name of the concept to the external buffer E1 where it was acquired by the experimenter routine E94 for designation.

Procedure D2 which followed Z5 attached the experimenter's designation to the concept. Process Q11 recalled the experimenter's designation and set a flag (H5) to indicate whether or not the concept had been attained. In this particular version we did not program any reaction to an incorrect concept because the mechanisms above insured that a concept was obtained with a minimum number of card choices and that incorrect concepts were never presented.

This particular version of the program elicited considerable discussion of the relative merits of storing items of information in short-term memory and then attaching a description to it using a P60-type routine vs. attaching the description to the information while it was in working memory and then transmitting the name of the information to short-term memory with a P70-type routine. It appeared that in the program both versions had been implemented within the various Z's.

Although the flow chart books do not seem to indicate its basis, there appeared to be yet a third type of processing within the working memory list. In some cases descriptive lists were attached to information while it was in M1 and then the name of the information was put on a list in the short-term memory. In other situations the description was attached to the name in working memory and the assumption was made that because the symbolization was the same, the information had also been attached in the short-term memory list. Here again it appeared as though we were stumbling around in handling memory. It should be noted, however, that the P60 routine was a generalized memory routine for attaching descriptions to information stored in memory, and P70 was a generalized routine for storing the name of information on the bodies of various lists.

CASE Mark III, Mod 1 was written up as a paper [Baker, 1965] presented at the Fall Joint Computer Conference in 1965. The following is an extract from the summary and conclusions part of that paper.

> When one reviews the history of the CASE program, it becomes quite clear that a subtle process is in effect, namely, as one's understanding of the learning process increases, the computer simulation program changes from routines which perform a large block of concept-attainment processes to a number of short routines which can be widely employed. In the CASE program, such a change has been dramatic at the P/Q level from Mark I, Mod 0 to Mark III, Mod I. The cynic will counter that we are merely learning how to code IPL-V, but I do not believe this is the only basis, as the change has been effected primarily on grounds other than coding considerations. The character of the sub-routines in the CASE program had also changed from being highly specific to a Bruner-type experimental situation to being reasonably independent of the experimental situation. They were, however, dependent upon the basic memory structure as defined earlier. The situationally dependent tasks still were performed, but the computer program was problem specific at a higher level than was previously true.

OUTCOMES OF THE CASE PROGRAM

There have been a number of outcomes of the CASE effort which are as follows:

1. The hierarchical structure of the processing routines and what appeared to be a parallel structure of context routines had led us into a continual search for logical units within the learning process. Behaviors which once seemed quite dissimilar had been decomposed and found to share a number of basic information processing modules. As a result we were slowly acquiring a better understanding of the psychological processes involved in concept learning.

2. The development of the CASE program had generated ideas for classical psychological experiments in a number of areas as a result of problems arising during the development of certain sub-routines.

3. A completely unexpected outcome was that we rarely made a production run on the computer, a fact which seemed anomalous in a computer simulation project. What had happened was that enormous numbers of man-hours had been devoted to gathering and studying protocols, to development of programming techniques in order to implement the next level of sophistication within the system, and to analysis of a computer program itself. These activities, plus a lack of variability in the learning behavior of the CASE program resulted in relatively few production runs.

CASE MARK III, MOD 2, DATED NOVEMBER 1965

In this version of the concept-attainment program the staff tried to see how much variability observed in the human protocols could be reproduced by rearranging the various routines in Mark III, Mod 1 and by creating a few additional P level routines associated with the wholist strategy. Much of the concern in the development of the Mark III version of the program had been in terms of interchangeability of routines, in other words with being able to rearrange routines without destroying the flow of information within the computer program itself. From the analysis of the protocols, four major types of behavior were selected which possibly could be simulated with the current edition of the concept-attainment programs and the addition of a few routines.

Conservative-Focusing Strategy

The first of these types of behavior was called S2, a conservative-focusing strategy in which the subject used a working hypothesis consisting of fewer dimensions than possessed by the focus object. Mark III, Mod 1 used the total focus object with just one dimension varied to create a search criterion through which object choices were made. In the protocols it was quite obvious that a number of subjects were using less than the total number of dimensions, so that it was of interest to see whether the computer program could be made to handle less than the total number of dimensions in its solution. In order to implement the S2 conservative-focusing strategy, a new procedural routine called Z7 was inserted between Z0 and Z1. The Z7 was called an initialization routine and established some characteristics of the subject prior to entering the main body of the strategy itself. Z7 consisted basically of a routine called P191 which would select K96 of the dimensions from the focus object in order to establish a working hypothesis. The working hypothesis would then replace the focus object and serve as the basis for selecting dimension values throughout the rest of the computer program. Because the computer program could vary all the dimensions on a working hypothesis and still not obtain a correct concept, it was necessary to introduce the Z6 routine, which was the subject's reaction to the experimenter's designation of a concept, at the end of the strategy list. In Z6 two routines were used: Q41 which inspected the list M13 and ascertained which dimensions had not been used and P181 which received the unused dimensions via the working memory and added them to the working hypothesis. When the program returned to creating a search criterion, it had at least one more dimension on the working hypothesis than it had initially. The Z1 routine then varied a new dimension and tried to create a search criterion involving it.

Wholist Strategy

The next strategy attempted was the wholist which was constructed by making minor modifications to the conservative-focusing strategy. Whereas in Mark I random number generators had been used to create the working hypothesis from the dimension values of the focus object, P191 in Z7 was used to create a working hypothesis of less than full dimension and then the ordinary Z1 routine was used to vary a single dimension of this working hypothesis. Thus, the search criteria differed each time the program used the Z1 routine. The procedure was well grounded in the protocols themselves. Analysis of the objects chosen and the reasons given for the choices showed that many subjects created a tentative hypothesis and then used a conservative-focusing type strategy to investigate each dimension of the hypothesis. The wholist portion of the strategy appeared in the use of the yes and no object designation information. In the wholist strategy a new routine called P211, which formed the intersection of all the yes objects, was used. The intersection formed by the yes objects was considered to be the concept and was placed at the top of the hypothesis list M11. The concept resulting from the intersection was then presented to the experimenter for designation. If the concept was not correct, procedure D1 checked the dimension values of the working hypothesis to determine whether all dimensions had been varied. If the dimensions had not been varied, the program merely returned to Z1 where a different dimension of the working hypothesis was varied. If all dimensions on the working hypothesis had been varied and the intersection did not yield the correct concept, the Z6 routine created for the new conservative-focusing strategy, S2, was used in order to find the dimension which had not yet been involved in a search criterion. The missing dimension was then added via Q41 and P181 to the working hypothesis and the procedure started over again. It was extremely easy to construct a wholist strategy given the initial basic processing routines of the conservative-focusing strategy. Even the routine P91 used by the conservative focuser to mark dimensions as relevant or irrelevant was used by the wholist strategy. The relevancy factor was not of interest here, only the involvement of a dimension value. The employment of P91 enabled the program to use all possible combinations of the dimension values as required by the wholist strategy. If one were unable to use all combinations, it was possible to miss the proper concept and the unusual application of P91 circumvented this possibility.

Gambling-Focusing Strategy

The S4 strategy was a gambling-focusing strategy in which two dimensions at a time were varied. The strategy involved a very minor change to Z1; a routine called Q31 was inserted after P131. An input constant K98 to Q31 indicated the number of dimensions to be varied. Thus, the program would execute P131 as many times as necessary to put K98 dimensions on the dummy description list. The remainder of the routines in Z1 were unaffected as both P141 and P151 would process all the information they received on the dummy description list containing the "from-to" information.

If the object choice was designated by the experimenter as not a member of the set defined by the concept, no information was gained by the subject and the computer program merely returned to creating a new search criterion which varied in only one dimension. But prior to varying any dimension a routine called P220 was used to set the probability values of the dominant dimension attribute values to positive. Again this was a function of our mechanical handling of the dominance features: a used dimension was set negative to prevent its being used again. In the present situation it was necessary to utilize both dimensions over again because no information was gained.

If an object chosen was designated as a member of the set by the experimenter, both of the dimension values varied were irrelevant and the ordinary P91 routine would mark them irrelevant. P90 had been rewritten in such a way that it would handle all of the dimensions named in a dummy description list describing a search criterion so that no changes were necessary within this routine to accomplish a gambling strategy.

Conservative-Focusing with Unintentional Variation of Two Dimensions

The last strategy involved was S5 in which the objects chosen would vary more than one dimension from the focus object, as in the gambling-focusing strategy, although the subjects believed they had varied only one dimension. The type of behavior modeled was commonly observed in the protocols based upon the original Bruner-type experimental materials in which the dimensions were psychologically dependent. A large number of subjects would make object choices in which more than one dimension was varied, yet when they processed the experimenter's designation of the object, they would only consider the more dominant of the two dimensions involved. The procedure used in the computer program to generate this behavior was exactly the same as that of the gambling-focusing strategy except that P91 was replaced by a new routine called P80. The new routine P80 would process only the first dimension instead of processing all the information on the dummy description list of a search criterion. The rest of the conservative-focusing strategy was as described under strategy S2.

The success of Mark III, Mod 2 was especially encouraging to the staff of the research project as most of the variability observed in the human protocols was reconstructed with the addition of only seven P routines to the total repertoire of routines. The routines at the P/Q level retained from Mark III, Mod 2 for future use were P191 which permitted work with less than the full focus object, Q41 which located untested dimensions, and Q181 which returned unused dimensions to the working hypothesis. The analysis of the four different strategies involved in Mark III, Mod 2 enabled the investigators to see that a number of P's could quite easily be generalized by using the K96, K97, K98, and K99 constants and by letting P90 use the dummy description list to determine what it should do. Thus, a number of the special purpose routines in Mark III, Mod 2 were later replaced by constants and minor modifications to already existing programs.

In the Mark III, Mod 2 program Mr. Martin had restructured the strategy list in a somewhat unusual fashion in order to make it more compatible with that of the list structure in IPL-V. At the time this was done, it appeared clumsy and the next version of the program did not utilize the Mark III, Mod 2 list structure. However, when the construction of Mark IV, Mod 2 was envisioned, the structure of the strategy lists had its basis in these structures.

The Mark III, Mod 2 version of the program was also a convincing demonstration of the ease of reshuffling existing programs without having to re-do the basic processes involved. For example, the construction of the various conservative-focusing strategies, even the wholist strategy, did not require any changes in memory structure, communication devices, or the interpreter, which was extremely encouraging. One of the ultimate goals of the project was to have computer programs which were interchangeable and freed from difficulties associated with the internal bookkeeping problems of the information-processing language in which the program had been written. From this point of view, Mark III, Mod 2 was one of the most successful of the programs in the current series.

CASE MARK IV, MOD 0, DATED 15 DECEMBER 1965

During the latter half of the summer of 1965, the present author spent several weeks at System Development Corporation. During conversations with Dr. Ross Quillian of Carnegie Tech., it became obvious that the short-term memory structure was similar to a very extensive net-type memory which he had been developing. In Dr. Quillian's memory structure, based upon the dictionary definition of words, one could define a word using a series of other words, and then, as each of these words was defined in turn, the original word would appear again on the memory list. The result was essentially a memory without a beginning and without an end. No matter where one started, the total memory looked like a gigantic description tree with any given point as its beginning. During the conversations with Dr. Quillian it became clear that the type of memory he was developing was quite similar to the memory lists in Mark III. Although the objects presented were on list M12, the description of how this object was found was also the description of the search criterion stored on list M11. The body of the object named on list M12 contained dimension values which were also contained and described on list M13. A rather highly inter-connected net had been constructed without formal intention. After having seen the close correspondence between the Quillian memory structure and the one already existing in our computer program, the staff decided to deliberately construct a memory structure which in some sense had no beginning and no end. The resultant memory has been designated as a "circular memory structure" (CMS). The basic idea was that, no matter where one started, a given entry point functioned as the top of a tree structure. Because the modular memory structure had been previously developed and memory procedures for handling this type of memory structure had been programmed, the circular memory structure was constructed with the basic memory modules currently in the program. The major change resulting from the circular memory structure was in where data were stored, not how they were stored.

In order to effect the circular memory structure, two types of lists were defined. The first was a non-bodied list, such as X11 9-3 O, which served only to hold a description list possessing specific attributes. Such a list had long been used in our program as a dummy description list created by U20 and stored through the P60-type routines. The attributes of the description list held by the symbol representing a non-bodied list could contain only specific attributes, as deep a level as one could go in the memory structure. If one did not establish some terminal point for obtaining usable information, it became impossible for the computer program to cease searching for information. Therefore, although the memory structure was circular, it also had a lowest level containing specific items of information. The memory processing routines did not change information on the non-bodied lists, but if new information was created it was put into a new dummy description list. For example, a given object might be designated one or more times by the experimenter, and each designation would appear on a separate non-bodied list. Another restriction was placed upon the non-bodied lists, namely that they could not serve as entry points into the memory.

The second type of list was a bodied list in which symbols were left on the body of the list within working memory by one of the memory processes. For example, in the previous program, P70-type routines would store a symbol on the body of a list. This concept had been retained, except that unique memory lists no longer existed and the symbol was stored on the value list of a particular attribute. For example, instead of having the list M11, a search-criterion attribute existed which described the working hypothesis. The symbols on a bodied list could serve as memory entry points. In addition, the information on a bodied list could be used as information by the P level processes of the program.

One can summarize the rules governing the circular memory structure as follows:

1. If a list had a body, its attributes were class attributes.
2. If a list had no body, its attributes were specific attributes whose value lists contained specific information.
3. If a list had a body, the symbols on the list could be treated as information.
4. The value list of a class attribute could follow rules 1 through 3.
5. Any list having a body could be an entry point into the circular memory structure for purposes of searching memory.

In talking about the memory net he had developed, Dr. Quillian reported that one of the problems which he was unable to solve, and which remains unsolved, was locating an initial entry point into the memory. Human beings are extremely facile about recalling the proper information at the proper time, but in a computer program this has been an insurmountable problem. To partially circumvent this problem a special register, the memory entry point, was created within short-term memory to describe the entry point in the circular memory structure through which current information was being extracted or stored. The memory entry point usually contained the same information as the working memory, M1. However, the distinction between M1 and the memory entry point was quite obvious when one considered that the working memory was one element deep and had no storage capability other than current information, whereas the memory entry point was a push-down list in which a trace of the previous entries into the memory were retained. If information previously acquired was necessary, one could merely reverse the list and obtain the previous memory entry points. Without this feature, one had to hand code each memory access as in Mark I, Mod 0.

The generalized memory processing routines, P60 and P70, were rewritten at this point to include the contents of the memory entry point as an entering parameter. Both P60 and P70 were extensively revised to take into account the nature of the circular memory structure. P60 could now be used whether the name in M1-N or the description in M1-D was remembered, whereas previously P60 could only attach descriptions. With the new circular memory structure it was also necessary to design a new recall-type routine, as previous programs had somewhat peculiar methods of recalling. The P level routine P500 was created to obtain the top symbol on the value list of a class attribute describing the list named in the memory entry point. To recall, for example, the experimenter designation of an object, one merely needed to have the name of the experimenter designation attribute and the contents of memory entry point; P500 would then search to the proper level and recall the designation. The sequence of search can be seen in the structure presented in Figure 3, Chapter II.

Inspection of the circular memory structure indicated two main domains that were not mutually exclusive: one was the acquired information net containing objects, dimensions, hypotheses, etc., and the other was the behavior net containing strategies, P's, Q's, etc. It should be noted that a deliberate attempt has been made to store the program as if it were data. The strategy itself was stored as a bodied list within the circular memory structure, and it was highly inter-linked to the information net through common descriptions of dimensions, dimension values, objects, objects found, etc. Even though the two branches of the memory structure tended to grow independently, there was, nonetheless, an extremely heavy overlap between them.

In the Mark IV, Mod 0 version the contexters' position in the model was indicated, but again they were not implemented. In addition, the marking of the dominant dimension attribute values as plus or minus when varying dimension values was eliminated. It was finally decided that dominance was a characteristic of the subject rather than something which needed to be programmed. Because of this, the P21 routine in the procedure Z0 was reprogrammed to create a copy of the focus object and place the dimension values of the copy in dominant dimension order. Thus, routines such as P131 and P141 no longer needed to use the probabilistic selection instruction J16 of the IPL-V language. They merely took the top dimension value as the most dominant, the second value as the next most dominant, etc. and accomplished what was previously done with an extremely complex set of housekeeping routines.

In the analysis of the Mark III, Mod 2 computer program, it became quite obvious that both the subject and experimenter should not use the same name for an object in the external environment. What an experimenter calls an object is not necessarily what the subject calls the object, and one of the important aspects of concept attainment is to learn names. In the previous editions of the program, when a description was attached to an object having the same name in both the subject's lists and in the external environment, experimenter access to the subject's internal representations and internal descriptions was implied. Such obviously was not the case. Therefore, the programs which received information from the external world were modified to copy incoming information and store a copy rather than the original in working memory. For example, in the external world, objects on the board were represented by the symbols 01 through 072, but a subject had to create his own label, normally an internal IPL-V symbol, to represent the object. Although this distinction in the representation of information appeared quite late in the computer program, it would have been an important distinction to have made earlier in the project.

The removal of the probabilistic selection device from the routines within the major portion of the strategy had considerable influence upon the internal structure of several routines. For example, P131 no longer did any probabilistic selection as it selected only the first K98 dimensions on the working hypothesis list and put them on the dummy description list. A major simplification resulted from the elimination of a loop involving the Q31 routine. In addition, the process P171 which reverted dimensions would process all dimensions it received on the "from-to list" provided by P90, but no longer had to restore dominant dimension attribute values to positive numbers.

The Z6 routine, which was the subject's reaction to the experimenter's designation of a concept, was not used prior to Mark III, Mod 2. In Mark IV, Mod 0 the procedure Z6 consisted of the Q41 routine which determined the number of untested dimensions and the P181 routine which added them to the working hypothesis. Both Q41 and P181 utilized the constant K97 which indicated how many dimensions were to be added to the working hypothesis.

In the flow chart book for Mark IV, Mod 0, there were a series of notes on trying to put some structure into the attribute system which had been in existence since the very early days of the program. The idea was to store the attributes in the modular format used throughout the rest of the memory and have them describe short-term memory. Each of the class attributes would be described in terms of the specific attributes, and the specific attributes would be described in terms of a few basic descriptive attributes. The idea was not programmed because Mr. Martin felt that as the class attributes had not been used in the program there was not great need for establishing them in a complex net. It appeared obvious that at some point in time there would be an attribute net which was intermixed with the current behavior and informational net, but it has not been designed. Hopefully future editions will make better use of the class attributes.

The structure of the memory net had suggested to the author that a decay-type memory might be nothing more than a breaking of links in a highly complicated memory net. Some psychological data indicated that people forget information under one set of circumstances yet recall it under another, suggesting that only certain directional links, rather than all connections to the information, have been broken. An interference memory might possibly be the existence of too many links in the memory net; information had been linked together that belonged together only under certain circumstances. The existence of excessive links within the memory net might lead one to the wrong information and give one the effect of interference. Both decay and interference were feasible to implement and suggested some research projects for the future. However, at the current time no programming effort has been devoted to these ideas.

In Mark IV, Mod 0 the best of the conservative-focusing routines created in Mark III, Mod 2 had been saved, and certain of them had been modified to accept numeric constants to control the number of dimensions worked with at various points within the problem. In that these constants have a specific influence on behavior, it may be possible at some later point in time to replace the constants with routines to create the same effect. Additional new features introduced in Mark IV, Mod 0 were the circular memory structure and the memory entry point. New P routines, namely P60 and P500, were written to handle memory processes in the modular circular memory. From a mechanical point of view a very important improvement in the programming was the elimination of all probabilistic selection devices by putting a copy of the focus object into dominant dimension value order. It seemed like a very simple thing to do, but it had major implications for many of the routines involved in hypothesis generation and reaction to designation of a concept. The programmer felt that a needlessly complex area of the program was simplified through eliminating the probabilistic selection mechanism.

An additional clean-up of the program was made by changing all P level routines so that they made references to memory only via the input names given to them by the interpreter. Prior to this, despite the staff's best intentions, P's existed which could reference memory locations not given on their input list, and this was highly unsatisfactory. The final innovation in the Mark IV, Mod 0 program was making the distinction between the name of information in the external world and the subject's internal name for the same piece of information, thus separating the last links which existed between external description and internal description of information.

CASE MARK IV, MOD I, DATED 6 JANUARY 1966

Throughout the flow chart books for the Mark III and Mark IV programs were references to contexting routines of one type or another. In the early Mark III versions low level contexters which established the input lists for the P level routines were mentioned and in the Mark IV, Mod 0 book there were some references to more elaborate contexting routines. From studying the protocols, it also became rather obvious that unless one made a major step in the so-called "backward strategy," the project would become fixated upon a detailed study of the existing program without making much progress along more psychological lines. In a series of discussions between Mr. Martin and the author, the decision was made to develop the backward approach on a more longitudinal basis instead of one level at a time within the concept-attainment process. It was decided that one should make a rudimentary model of the total concept-attainment process rather than developing an elaborate model of only the strategy aspect of the problem. The decision was somewhat radical because up to this time all of the emphasis had been upon strategy and very little upon the creation of strategy by the subject or his manipulation of it. Much of the discussion of the concepts underlying the contexting routines has been presented in a previous chapter and the attempt to develop the contexter idea is described in the paragraphs below.

In Mark IV, Mod 1 a higher level strategy list (S3) implementing what one might call the central executive or the supervisory aspects of concept attainment was created. The body of the S3 list contained routines C10 which created a skeleton strategy from the experimenter's instructions, C11 which filled in the skeleton strategy on the basis of past experience, T60 which gave the created strategy list to the interpreter for execution, a post-mortem analysis routine, C12, and X20 which was a final post-mortem print routine to record what happened in the solution of the problem. The major focus of the contexting operations was to devise a system whereby one could create a strategy from a pool of existing Z and P level processes. Such a goal immediately brought into focus the problem of describing behavior. Preparing a description of behavior so that it can be utilized by a computer program is an extremely complex task and the staff does not pretend to have any clear understanding of the process. The current descriptive scheme was an extension of the earlier attribute system in use for quite some time in the project. Very early in the project a P level routine was described by the attributes A1 for input, A2 for output, and A3 for process description. The Mark IV, Mod 1 version employed the not previously used value list of A3 to add a non-bodied list whose description list contained up to four specific attributes—A30, A31, A32, and A33—describing the routine. The value list of A30 described the type of routine, namely an operation, a decision, or a contexter; A31 was the communication mode whose values were to external world, from external world, and both; A32 was an information-created attribute whose values indicated the search criterion, object choice, concept, focus object, and working hypothesis; and A33 was the designation attribute whose values indicated whether an object or a concept was designated. With these four attributes and their values it was possible to uniquely describe each of the Z level routines existing in the Mark IV, Mod 0 version of the computer program.

Because each of the major procedures at the Z level in the strategy list corresponded to some particular behavior of the subject, it was thought possible to describe the experiment in terms of the above-mentioned attributes. The experimenter's verbal messages to the subject were coded using various combinations of the descriptive attributes and their values. The messages were then stored on a list named L60. An attribute of each message described the type of message by means of a set of four attribute values. Thus, the contexting computer programs could examine the message and ascertain what type of processing needed to be done to the information it contained. For example, a typical message occurs when the experimenter tells the subject to select an object. The procedure defined by "select an object" could be described as a doing routine involving the search criterion, and the message itself described as a process specification. The routine corresponding to this message was procedure Z1, which created a search criterion. In the next message on the list the experimenter said, "From the board in front of you." This message was described as a procedure involving doing and communication from the external world, where the information from the external world was the object choice. Such a scheme was a very rudimentary method for getting around semantic and syntactical analysis of English; however, for the immediate purposes of the present program it appeared to be reasonably adequate. Other types of messages provided the name of the external environment, the relationships between dimensions and their dimension values, the type of concept to be learned, and the goal. At some further date in the development of the program, hopefully this message scheme will be made more general, thus getting around the very narrowly defined attributes which are highly situationally dependent. Doing so depends upon obtaining a much better understanding of describing behavior. The flow chart for the Mark IV, Mod 1 program showing all levels of contexters is presented in Figure 6.

The routine which handled the translation of experimenter messages into a skeleton strategy was called C10 on the higher level contexting list. C10 consisted essentially of an executive program which received the experimenter messages, created a copy of the descriptive portion, and attached this description to a symbol on the skeleton strategy list. The remaining routines within C10 stored in short-term memory the description of the problem to be solved and the various types of experimenter designation information. The output of C10 was a list whose symbols carried descriptions matching Z routines, although these symbols were not filled out in terms of P's and Q's. It was the function of routine C11, consisting of routines C111, C112, C113, and C114 to fill in the skeleton strategy list by matching descriptions of desired behavior given on the skeleton strategy list with those of recently executed behavior on a strategy list in the long-term memory. As C111 found routines in long-term memory matching the description desired through the use of an R level sub-routine, it also checked to see whether external information was obtained by the Z routine located. If so, routine C112 would insert the appropriate contexter after the Z routine on the strategy list. Upon completion of the translation of a skeleton strategy into an executable strategy, routine C113 put in the proper links. For example, each time there was a decision-type routine, two links would be inserted, one for the no branch and one for the yes branch. In each case, the no branch returned to Z1, which was the hypothesis creation routine, and the yes branch proceeded to the next symbol on the strategy list. At the end of C113 an executable strategy called the new strategy list existed in working memory. Routine C114, which was the last one within C11, would then place the new strategy list on the value list of the strategy attribute of the problem list L100 in short-term memory. It would also place the name of the strategy in the memory entry point.

The basic purpose of C10 and C11 was to translate the experimenter's verbal description of the problem to be solved into a skeleton strategy and then to convert the skeleton strategy into an executable strategy list called the new strategy list. The new strategy list possessed contexters C21 to create the initialization routine Z7; C37 to create the analysis of the object choice; C38 to create Z4, the subject's reaction to the designation of an object; and C22 to construct the Z6 routine which reacted to the designation of the concept. The details of the contexters at the C20 and C30 levels will not be described here because the current versions have been explained adequately in previous chapters.

At the time Mark IV, Mod 1 was written, there was considerable confusion on the part of the author as to where one does the decision making which results in the within-problem variability exhibited by subjects. The approach taken in Mark IV, Mod 1 was to create all possible branches within the contexters themselves and to utilize the within-subject descriptive information such as the awareness factor and the parameters K96, K97, and K98 to construct a very large complicated decision net. Because of that approach, the C20 and C30 level routines became large, extremely clumsy, and quite difficult to program in any reasonable fashion. The approach resulted from confusing the characteristics of the subject with the task of contexting, and at that point the author was not aware of the distinction.

Mr. Martin programmed Mark IV, Mod 1 in IPL-V. However, it turned out to be somewhat of a "super-kludge." He attempted valiantly to put some order into the chaos, but the conception of what the contexter should do and how it should proceed was extremely poor. The resulting program appeared hopeless, and

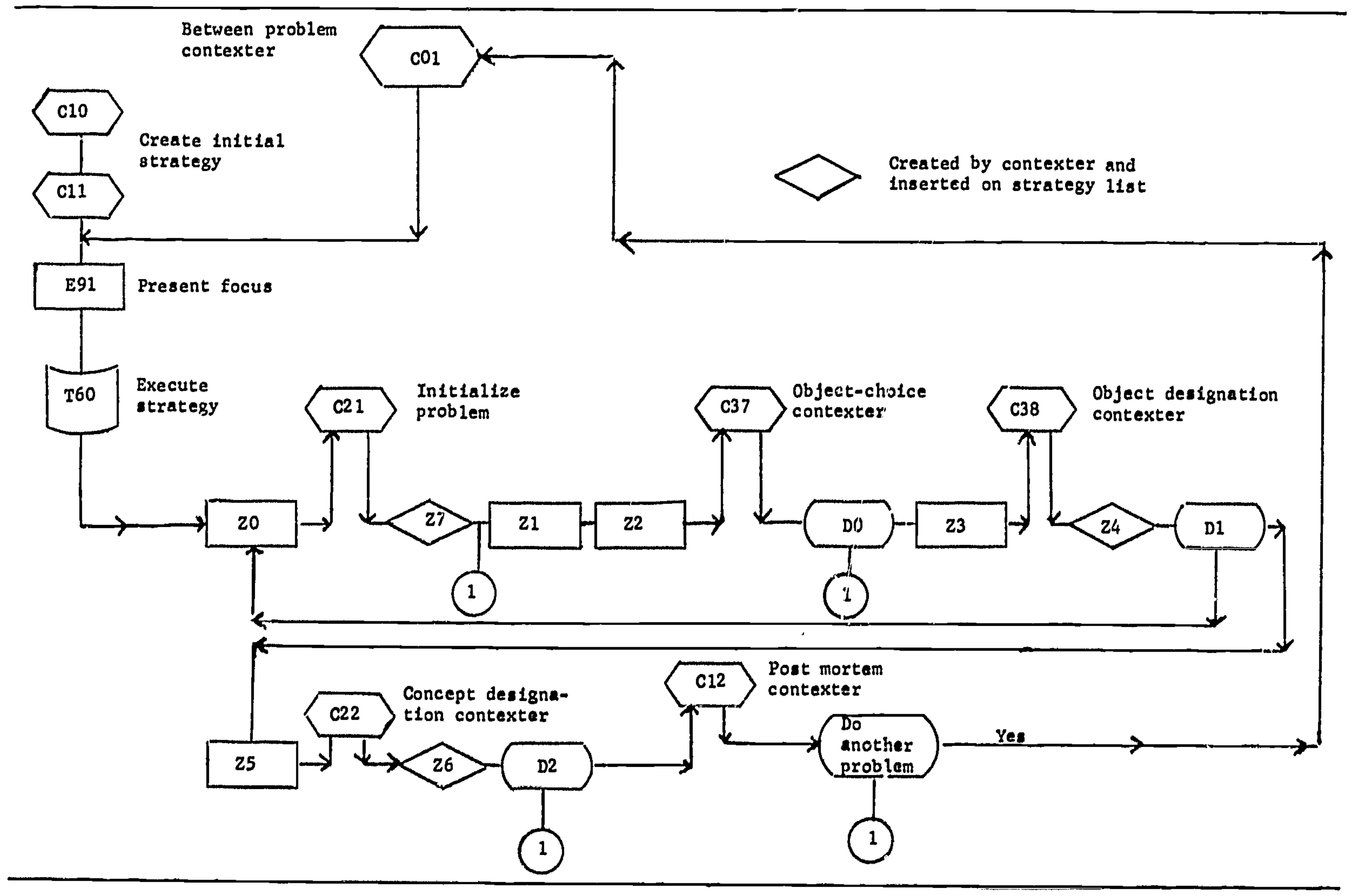

Fig. 6. Flow Chart for Mark IV, Mod 1 Containing All Levels of Contexting Routines

debugging was terminated once the IPL-V code had been written. On April 18, 1966, it was suddenly realized that Mr. Martin's previous work done in Mark III, Mod 2, creating a small number of steps in a list structure format, would enable development of what was called a ' single contexter." The single contexter was a high level routine which would create a step of the program, execute it, and then perform a contexting operation to see what the results of the step were. Although it was not truly a single contexter, the idea was labeled as such. The insight achieved at this point served as the basis for the Mark IV, Mod 2 version which has been described earlier.

In addition to Mark IV, Mod 1 being conceptually confused, many of its mechanical aspects were badly structured. The experimenter messages had not been standardized following the modular memory structure; the contexting routines were not programmed following the pseudo-code scheme; and many branches of the complex decision nets within the contexters were not feasible to program at all. Thus, in Mark IV, Mod 2 the mechanical aspects of the contexting routines were completely redesigned and, as had been the case with the strategy lists earlier, a significant effort was made to bring this new program into conformity with the previous programs. The final version of the program, reported in the earlier chapters of this book, was essentially an extremely redesigned, repackaged, refractionated version of Mark IV, Mod 1. Many of the finer programming details were retained from one version to the other; however, the major structure of the contexters was completely redone after discovering the difficulties in the initial version of the contexters.

SUMMARY OF THE CHAPTER

During the development of the various concept-attainment programs, a number of major themes evolved, some rather early in the project, others only after a number of attempts to model the concept-attainment process. The original computer program, Mark I, was based upon a rather intuitive idea about how the author would solve the concept-attainment problem. In attempting to write an IPL-V program for the concept-attainment task, it was necessary to introduce things such as random number generators to create hypotheses and record-keeping systems for determining which possible combinations of dimension values had been used. The program reflected neither a clear-cut underlying strategy nor a clear-cut understanding of the underlying mechanisms. Mark I was just an attempt to see whether a program could be written to attain a concept. In addition, an attempt was made to provide the program with a certain amount of variability in its object choice behavior through the use of various constants, length of lists, and other mechanisms of this general type. At the time the first program was written, it followed the basic approach underlying many of the published programs for various cognitive behaviors.

A number of lessons were learned from programming Mark I version of the program, and most of these were associated with learning to program in the IPL-V language. Because extensive sub-routining is standard practice in scientific programming, programmers have experience with fractionating problems and recognizing reasonable sub-routines. Such was not the case for the author in IPL-V as the procedures and processes involved were relatively new and appropriate fractionation of the problem was not readily apparent. In the original straight-line program no serious attempt was made at sub-routining. However, the resulting program quickly showed that one needed to be much more careful about sub-routining in simulation programs than in scientific programs and much of the later effort of the project was devoted to a continual fractionation process. Although Mark I was not very sophisticated, it clearly demonstrated the feasibility of this type of programming to the present author and suggested a longer term project would be feasible.

The learning strategies suggested by Bruner et al. [1956] have served as a focal idea within the concept-attainment project, and the concept of a strategy list appeared very early in the development of the system. Although such a strategy list was not used in the original version, the flow charting books prepared during the summer of 1964 developed the strategy list and a symbolic representation of procedures, processes, and information processing modules. The only features of the strategy list idea that have changed very much over a period of time are some of the mechanical aspects such as the number of links following a decision point. A number of different schemes were proposed to implement the actual execution of the strategy lists, and eventually Mr. Martin developed an interpreter program, an extremely sophisticated IPL-V program. The interpreter developed in late 1964 to execute the strategy list remained unchanged through Mark IV, Mod 0. When the high-level contexting operations were introduced in Mark IV, Mod 1, it became necessary to make minor modifications in the interpreter in order to identify entry into a contexting routine.

One can observe in the descriptions of the various Marks and Mods of the program a rather subtle change in the nature of the routines at the Z and P level. In the early days the Z's and P's corresponded to rather large segments of the concept-attainment process, and it was necessary to continually redefine each of these symbols. The reader should not be misled by the fact that Z1, Z2, Z3, etc. have been used since the earliest days because the routines these symbols represent have changed very radically. There have been essentially three major restructurings of the strategy lists and hence of the program itself. The first of these occurred at Mark III, Mod 1 after it was discovered that the several memory process routines were identical except for the inputs. A major effort was then made to find communalities throughout the program and utilize the same processes in several different situations. The second major restructuring of the program occurred in Mark IV, Mod 0 when the circular memory structure was introduced. All of the memory processing routines, and a number of other routines, were redesigned to take into account the incorporation of the circular memory structure and the memory entry point in the program. The third major restructuring of the program occurred in Mark IV, Mod 2, where the contexting routines were introduced at three levels. The first level contexting routines, C10 and C61, created the strategy. Both the second level contexter, C22, and the third level contexters, C37 and C38, created Z level routines which were situationally dependent.

The fractionation process is by no means complete. It can be seen quite readily in Mark IV, Mod 2 that the working-memory processes need to be restructured and some subroutines developed to handle the transfer of information to and from working memory. Such routines have been designed but have not been programmed. The P's and Q's in the current version are still too large and the amount of information processing they do is too extensive. A further fractionation of these routines depends upon more information about human cognitive behavior than is currently either available in the psychological literature or observable in the protocols.

One of the basic tenets of the current program was that of the "backwards" approach, in which one started from a program for a very experienced subject and tried to work backwards to a subject who is less experienced in solving concept-attainment problems. Through Mark III, Mod 1, the computer model was strictly that of an experienced subject. In Mark III, Mod 2, it was discovered that with relatively little effort nearly all of the basic types of variability observed within the protocols could be created by assembling various P's and Q's into new types of Z's. For example, variations within the conservative-focusing strategy have been introduced principally through the means of the constants K96, K97, and K98, although in the Mark III, Mod 2 version they were handled somewhat clumsily. In the Mark IV, Mod 2 version the three constants elicit all of the variability, other than the wholist strategy, previously observed in Mark III, Mod 2. In Mark IV, Mod 1, the awareness factor idea, which was related to the psychological dependence of the dimensions in Bruner-type materials, was also introduced. A considerable amount of variability can be constructed through the awareness factor. Its psychological origins are considerably deeper, but the parameter is a reminder to look at this type of behavior. At the current time, the within-problem variability exhibited by the computer program is quite satisfactory, but it is unfortunate that such variability results from the "screwdriver" parameters, K96, K97, K98, and the awareness factor. The ultimate goal is to have the within-problem variability result from the "subject's" own mechanisms. Eventually, the within-problem variability will occur at the contexting level where, through misanalysis or other mechanisms, the program will create its own variability. Such a capability is presently provided by having those of us on the outside of the program code it in through the "screwdriver" parameters. Internal creation of variability in behavior is not a trivial problem. Many other people have looked at this problem; answering it means that one has accomplished Newell's [1962] "solution by understanding," and this does not appear to be on the immediate horizon.

In retrospect, it appears that the major portion of this project's programming effort was devoted to memory structure. It was realized in the summer of 1964 that much of the success of the concept-attainment model would depend upon how adequately memory structures were modeled. In the original version of the program, no attempt was made to model memory. Information was merely stored in IPL-V lists and data terms, and the computer programs were written to extract necessary information from storage when it was needed. The first version of the program in which any serious attempt was made to build a memory model was Mark III, Mod 0, and in that program the three-level breakdown of working memory, short-term memory, and long-term memory was utilized. The two cell idea of the name and description within working memory was also invented. The mechanics were quite rudimentary and the idea of a dummy description list, although mentioned, was not fully developed. The Mark III version of the program also introduced the modular memory structure. The problem arose of determining when the program had reached a level at which information was available, and the non-bodied lists containing specific attributes were invoked in order to terminate the searching procedures. The Mark III version had a confused scheme for extracting information; some P's would use a memory process to acquire information whereas other P's would directly use the name of a list to obtain the information. The confusion reflects the investigators' hesitancy in structuring memory.

In the Mark III version of the program, it became quite clear that most of the information dealt with by the program was descriptions of other information. The modular memory structure was designed to implement storage of descriptions rather than storage of specific items on lists. Although a rather highly interlinked memory net was inadvertently developed, it was not until the discussions with Dr. Ross Quillian during the summer of 1965 that the possibility of completely interlinking the memory net was realized. With this concept in mind, the memory aspects of the program were completely redesigned in the Mark IV, Mod 0 version where the circular memory structure and the memory entry point were introduced. Although the circular memory structure was new, the modular structure utilized in Mark III was retained as the mechanisms were well understood and seemed to be functioning fairly well. The problem of computer program control of storing and recalling information is still unsolved and is one of the major unsolved problems in modeling cognitive behavior.

The subject's control of his own behavior, i.e., contexters, had its origins in the very early days of the project and, throughout all of the flow charting books marginal notes record various ideas about contexting. The original contexters were conceived of as low-level programs which would establish the input list under A1 and output list under A2 for each of the P routines, but the low-level contexters were never programmed due to structural difficulties in Mark III. After development of the circular memory structure in Mark IV, Mod 0, it became obvious that representation of the total concept-attainment process was necessary. It was decided to make a rudimentary model of the total contexting process from the experimenter's instructions to the actual execution of the program. Again the low-level contexters escaped attention, and computer programs to set up the inputs to the various P's have not been written, primarily because an adequate description of behavior is not available. The attribute system used in the experimenter messages to describe gross behavior and also to describe the procedures on the strategy list was merely a temporary device to be used until a better insight is gained. Newell's article [1962] on the internal organization of computer programs provides several examples of his attempts to resolve this problem within the General Problem-Solving program. Development of a system for describing behavior which a computer model can handle alone is an extremely difficult task and so far has eluded investigators involved in computer modeling of cognitive behavior.

The present chapter has attempted to provide the reader with some of the chronology of the development of a computer model of the concept-attainment process. It may appear that the discussions above are rather chaotic and disorganized, but they reflect the course of the project. One of the difficulties encountered was that many good ideas have come and gone because the author was not prepared at a particular time to see their import and at a later point in time was unable to recall them. The staff feels the level of sophistication of the computer model has risen as the number of Marks and Mods has increased, and hopefully the reader will concur.

IV
SUMMARY AND CONCLUSIONS

The model has been developed to its current state through a combination of protocol analysis, computer program analysis, and hours of spirited debate. A comparison of the first concept-attainment program with the current version reveals many differences, some obvious, some subtle, but, hopefully, all in the direction of increased understanding of the concept-attainment process. As was indicated in the introduction, the concept-attainment task was chosen because it appeared to be a simple task and easy to program. There was little realization that it would lead to a hierarchy of contexting routines, a model of memory, pseudo-code schemes, and many other facets of the present model. Each problem encountered, and the solution devised for it, merely served to expose previously hidden considerations which were more difficult and more important than the problems previously encountered. Thus, the deeper the project has delved into concept attainment, the more complex the psychological processes have become. The original estimation of the simplicity of the task has changed to respectful awe at the potential complexity of even the most rudimentary cognitive behavior. Such a new frame of reference has strongly reinforced the author's conviction that computer modeling provides a powerful tool for investigating cognitive behavior.

Because the previous chapters have discussed the several versions of the computer model in elaborate detail, no attempt will be made to summarize the various programs in this chapter. Rather, the present chapter will be devoted to discussing the salient features of what the staff feels was learned from its experiences in computer modeling of concept attainment.

MODELING CONSIDERATIONS

Internal vs. External Information

In the early days of the project, the concept-attainment process was thought to be primarily one of processing information received from the external world. However, in developing the computer program to the current point, it became apparent that the majority of the information processed does not come from the external world but is created internally by the subject. Thus, although concept attainment is an information-processing problem, the amount of external information processed is extremely minimal and consists only of the experimenter's instructions and his designation of object choices and of concepts. It should be noted that perception problems associated with observing dimensions and their values were intentionally omitted as is typical of most existing computer models. If the majority of information is created internally, it then becomes a task for psychologists to determine what internal information is created and how it is processed. For example, from a protocol it is quite easy to determine that when an object is designated as a yes or a no, the subject creates information about the relevancy or irrelevancy of a particular dimension or dimension value. If one is to develop an adequate computer model, it is necessary to know what information is created, on what basis a subject created the information, what he did with it, and how much of it was retained for longer term use. Without substantial knowledge of this type it becomes difficult to develop sophisticated computer models. Unfortunately the current techniques of psychological experimentation do not seem capable of providing the requisite insight.

The Memory Model

Analysis of the concept-attainment task indicated that it was virtually impossible to do any significant modeling of the concept-attainment process without coming face to face with some model of the structure of memory and of the cognitive processes associated with remembering and recalling. The three-level structural model of memory developed for the present simulation program appears to be a reasonable model. The idea of the working memory functioning as a temporary holding-type memory has proved to be an exceptionally useful concept as it enables information to be communicated from routine to routine without going through the rather complex mechanisms associated with short-term memory.

Conversations with Dr. Ross Quillian at System Development Corporation made the author recognize that the memory structures in the earlier editions of the concept-attainment program were very nearly memory nets. After having talked to Dr. Quillian about this particular problem, the investigators redesigned the memory to the present circular memory structure. The use of a list-structure format for memory has seemed excessively artificial to the present author and the circular memory structure appears to provide a reasonable alternative. The significant feature of the circular memory structure is that, although the memory processes in the model can store and recall information, the memory does not consist of a series of predefined bins into which information is automatically placed. The memory structure is dynamic in that storage is created in the proper structure as the information is created, rather than being specified ahead of time. The dynamic nature of the circular memory structure also gave rise to the problem of entering the memory structure and keeping track of location in memory. Because the order in which memory is created is situationally dependent, the memory entry point (MEP) has proven to be quite successful in performing the bookkeeping associated with the circular memory structure. The problem which is as yet unresolved is a mechanism for entering an existing memory structure, such as would be required when a second or subsequent concept-attainment problem was begun.

The memory model employed is somewhat clumsy mechanically; however, its structure does provide for the eventual inclusion of both interference and decay-type forgetting. The inclusion of forgetting in the computer model will again raise many more problems than it will solve but should prove to be of interest.

Attribute Structure

The Mark IV, Mod 2 computer model involves approximately 25 attributes under which various types of information can be stored. These attributes are divided into class attributes and specific attributes, and certain mechanics permit the computer program to ascertain what information is available under these attributes. For example, under a class attribute "chunks" of information are available; under a specific attribute unique items of information exist. The attributes employed are a function of the particular experimental situation modeled and represent an initial approach to the exceedingly difficult task of describing behavior. The next logical step appears to involve creating both class and specific attributes from a minimal set of basic descriptive attributes, but the logical basis for defining such a basic set of attributes is not presently obvious to the author. It does appear to be quite possible for the computer program to create both class and specific attributes when required by the situation. Hence, given a basic set of attributes, the computer model could handle the descriptive processes using its own capabilities. The attributes currently used were devised by the computer programmer and as such merely identify or label different units of information which he believes necessary. However, to shift this responsibility from the programmer to the computer program is the next major step, and one which clearly needs to be taken.

Use of Protocols

The "think-aloud" protocols, especially with experimenter interrogation of the subject, have been an excellent device for eliciting the grosser behaviors exhibited by subjects within a concept-attainment task. The protocols have been extremely disappointing in providing answers to the more fundamental questions. It seems as if the "state of the art" limitation in protocol analysis had been reached, and it would be difficult to elicit much more information from the protocols than was extracted. The failure of the protocols to provide answers to questions about the internal mechanisms of human subjects, such as contexting and memory, suggests that new types of psychological experiments are desperately needed.

Contexters

During the early phases of the present modeling project, the computer model consisted essentially of the strategy list with its Z routine, P routines, and Q routines. More detailed fractionation of the computer program itself revealed the necessity of separating the central executive function from the operational function. There are actually two processes which occur in parallel as a human being solves a problem. One was designated the contexting process which is the monitoring, supervising, goal-directing aspect of human behavior, i.e., the higher level cognitive processes. The second is the operational aspect involving what one might call the subject's abilities, habits, or mechanisms. Once the difference between the contexting program and the operational program had been conceptualized, a major restructuring of the computer program was possible and made for significant differences in the model of cognitive behavior.

A contexter may be viewed as creating a plan or strategy for behavior. At high levels in the model it creates plans for overall behavior and at low levels it creates plans for very specific actions. Such a planning hierarchy was first envisioned by Miller, Galanter, and Pribram [1960] when they suggested the existence of plans which create plans. The Miller, Galanter, and Pribram [1960] scheme and the present hierarchy of contexters have two implications for the internal organization of a computer model. First, the organization of the program must be such that it can treat itself as data; second, a contexter must be able to create programs from the "abilities," i.e., sub-routines possessed by the "subject." In the first case, the contexter routines must be able to analyze, modify, and otherwise manipulate the computer program itself. Without such a feature, the contexters cannot improve the "subject's" performance as a function of experience. The mechanics of treating the total program as data can be accomplished through interpreter schemes such as that programmed by Baker and Martin [1965] in which the strategy or plan is a list of symbols representing behaviors. However, the symbols are executed by means of an interpreter rather than directly in the underlying language. Because these symbols are placed on lists, they can be treated as data through the list processing language and be manipulated by the context routines. It should be pointed out that the Baker-Martin scheme divorces the contexting operations from the interpreter as the contexters are also executed by the interpreter.

The lack of differentiation between data and program means that both must share a common internal representation and that the internal organization of the computer program must facilitate both the storage and retrieval of information in some uniform fashion. In most existing computer models, the memory processes have been avoided by having the computer programmer remember where he stores the information and recall it for the program via the code he writes. Under an adequate computer model, the program should decide what should be stored and store it for retrieval through internal recall mechanisms. Uniformity of storage and retrieval in the present model has been implemented through a modular memory structure accompanied by basic remembering and recalling routines which are a function of the structure of the memory rather than the list processing language employed. However, the programmer still decides what to remember and when to recall the information.

In addition to devising a system through which the program can be manipulated, it is also necessary to provide contexters with the capability of creating new programs, based upon new generalizations inductively acquired; i.e., the contexters are programs which can create programs. Because the lowest level of detail in a computer model consists of basic processes which can be executed, i.e., the "abilities" possessed by the subject, all other levels of a computer model are composed of the symbols which represent these basic processes. Hence, the procedure for creating new processes, plans and contexters consists of restructuring these basic processes in an appropriate order. However, if the context routines are to have the capability of creating plans, must "know" or be able to ascertain the capabilities of the basic processes and of the higher level routines which derive from them. There is a crucial and as yet unresolved requirement for being able to describe the characteristics and capabilities of a behavior regardless of the level at which it appears in the computer model. One rudimentary way is to consider a process as a transformation and use its inputs and outputs to describe the nature of the transformation; however, Newell [1962] indicates this is not an adequate description. Regardless of how the description problem is solved, it is quite clear that unless it is solved, progress in computer models will be very slow. It would appear that Newell's "solution by understanding" requires a prior "solution by description."

Programming Techniques

A number of computer programming techniques have been developed by the project staff. The foremost of these techniques is the pseudo-code interpreter system which enables one to represent the model as a list of symbols. The pseudo-code scheme also provides the mechanical basis for the capability of the contexters to create programs from existing programs; a major unsolved task is the conceptual basis for such a capability.

The circular memory structure and its generalized remembering and recalling routines hopefully provide the basis for future computer programs which can perform these processes without human supervision. Again the mechanics have been provided but the requisite knowledge upon which to base the processes is not available. The modular structure of the circular memory also permits the learning strategy to be stored as if it were data and provides yet another small stepping stone toward computer programs which can create other computer programs.

The development of computer programs in which the program can be treated as data and new behavior sequences can be created requires that the computer model be independent of the mechanics of the language in which it is coded. In any programming language there are a large number of housekeeping tasks which are necessary to keep a program running, but which are unrelated to a computer model of cognitive behavior. For example, in IPL-V one must erase unneeded lists, push and pop the H0 Communication Cell, and make copies of lists. If the computer program is to truly be a model, it should not be cluttered by additional features which take account of the housekeeping details associated with the underlying programming language. Freedom from such mechanical details can be accomplished through the use of an interpretive system such as the pseudo-code in the Baker-Martin [1965] scheme. Alternately, if a "solution by description" were achieved, it could serve as the basis for the development of a compiler-level modeling language. One could then model the cognitive behavior in this language and be completely freed from the underlying list processing or other such language. Regardless of the method, the computer model must be freed from the housekeeping mechanics of the underlying programming language.

RESEARCH IDEAS GENERATED BY THE COMPUTER MODEL

1. In that the total computer model was developed around the idea of a strategy or plan, there exists a need for more information on what processes a subject uses to create plans and also to establish the role of instruction in forming such plans. As was observed above, the present computer program assumes the experimenter's instructions have a crucial role in the establishment of at least a gross plan of behavior. It would be very interesting to conduct some studies to ascertain whether or not subjects perceive instructions in this light and how they utilize the information in planning their approach to concept attainment.

2. When one considers the vast realm of behavior which human beings are capable of exhibiting, it is quite remarkable that in a given situation they normally produce behavior which is quite relevant to the problem at hand. It may not be effective in a given situation, but usually it has some possibility of being useful. One of the outstanding observations from the protocols was that almost all of the subjects very quickly produced a plan for solving concept-attainment problems. If subjects were not able to select behaviors rather rapidly and appropriately, they would require a much greater period of time to solve these types of problems than was observed. Therefore, an urgent area of research is that of how humans select a specific behavior from their repertoire of possible behaviors.

3. In that the communication between the experimenter and the subject is minimal in the concept-attainment experiments, it is somewhat unusual that subjects can maintain a sense of goal-directedness during the entire experiment, especially in the absence of much in the way of external clues. One needs to investigate quite carefully the relationship between what the subject sees as the task to be accomplished and what kinds of information he utilizes to ascertain whether he is making progress toward that goal. From analysis of the protocols, it was evident that most subjects had some understanding of whether or not they were going in the correct direction despite the lack of external clues. It would be very interesting to ascertain what types of internal information they were utilizing to maintain this goal-directedness.

4. The memory entry point which was created to maintain some sense of order in the circular memory structure raises many questions about how people store information and, more importantly, how they get it back once it has been stored. The nature or structure of information stored in the human brain is not intuitively obvious. Subjects are quite adept at getting the information at the proper time and proper place without visible effort. Logical analysis of the concept-attainment problem suggested that people followed some type of a memory entry point sequence in that they tend to remember information about what they are currently working with without much concern for the details of the previous operation.

5. Much of the effort in the past year was devoted to trying to introduce within-problem variability into the computer model. A lack of understanding of how people make errors severely hampered the process. Stimulus-response psychology has traditionally blamed

errors upon the stimulus material; however, our model tends to indicate that these errors are more likely due to errors in the contexting operations and internal description rather than in the stimulus materials themselves. Experiments designed to obtain some understanding of how humans make errors in the internal processing of data would be most helpful.

6. One of the large so-called "fudge" factors in the current program is the awareness flag, designed because the protocols showed that many subjects inadvertently worked with less than the full set of dimensions. In some cases it was clearly a perceptual problem, in other cases, it was possibly an oversight. If one asked a subject to name the dimensions, he would mention all five, yet in working on a given problem, he would deal with less than the full five dimensions. The behavior raises a question of how people decide upon using less than the full information and how they handle it when they work with less than the full information. There are two sides to this coin, one of which is when the subjects know they are using less than the full amount of information and the other is when they do not. The interesting facet in the latter case is why they don't know.

7. Analysis of the protocols indicated very clearly that people remember not object choices but strategies and that they reconstruct rather than recall. Such observations raise many questions about the roots of the concepts currently in vogue about memory and what is stored. The protocols gave a very distinct impression that people remember extremely little detailed information but do remember with great fidelity the strategies, procedures, and processes necessary to reconstruct the sequence of events. It appears that people keep detail around just long enough for it to be of some use. However, any information stored for a longer period of time is usually stored in the form of a procedure, i.e., a strategy accompanied by enough basic information to repeat the process itself. Such a conceptualization of memory enhances the idea of the working memory and of the short-term memory, where working memory keeps the details just long enough for them to be used and short-term memory keeps enough of the salient information so that the process can be continued. It would appear that the long-term memory is devoid of large amounts of detail, but contains strategy lists and the necessary and sufficient amounts of crucial information to execute the strategy. However, the mechanisms by which people reconstruct rather than recall are not obvious and they seem to be a good topic for future research.

8. During the development of the short-term memory, it was observed that the information was stored in a highly interlinked fashion, no matter what structure of memory was used. The existence of such a high level of interlinkage seemed to suggest that interference in memory could be caused by access to inappropriate information resulting from the excessive linkages of the stored data. It would be very interesting to perform some experiments in which one deliberately caused subjects to remember certain types of linkages and then observed the amount and nature of interference that occurred due to the preconstructed linkages.

The types of information that are required by the present computer model in order to develop it further along the lines indicated suggests a rather different realm of psychological research than is usually reported in the literature. The concern is with what the subject does rather than what the experimenter does. In most current psychological literature, the experimenter is actually varying the material, etc. and very little, other than some relatively gross outputs, is ever attributed to the subject. The protocol analyses have shown that these gross outputs are not really informative about the processes, procedures, etc. utilized by the subject. In essence what is needed is some research in depth as to what subjects <u>do</u> in experimental situations rather than what they <u>produce.</u>

THE STATE OF OUR ART

The concept-attainment program currently available, namely Mark IV, Mod 2, is a very rudimentary model of the concept-attainment process and by itself does not exhibit a great deal of what a specialist in simulation would call "interesting behavior." However, the author has not been overly concerned about this aspect as the computer program essentially represents a repository for ideas about the concept-attainment process acquired to date. From this point of view, the program can be considered quite successful in that a reasonable understanding of at least the grosser mechanics of the concept-attainment process was obtained, at least the project staff thinks it was. In the modeling of the concept-attainment process, many problems have not been solved, but the modeling process has provided quite a good idea of what problems need to be solved in order that further progress can be made.

REFERENCES

Baker, F. B. An IPL-V program for concept attainment. Educational and Psychological Measurement, 1964, 24, 119-127.

Baker, F. B. CASE: A program for simulation of concept learning. AFIPS Conference Proceedings, 1965, 27, Part 1. Pp. 979-984.

Baker, F. B., & Martin, T. J. An IPL-V technique for simulation programs. Educational and Psychological Measurement, 1965, 25, 859-865.

Brian, C. R., & Goodenough, F. L. Relative potency of color and form perception at various ages. Journal of Experimental Psychology, 1929, 12, 197-213.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. A study of thinking. New York: Wiley, 1956.

Hunt, E. B. Concept learning: An information processing problem. New York: Wiley, 1962.

Hunt, E. B., Marin, J., & Stone, P. J. Experiments in induction. New York: Wiley, 1966.

Johnson, E. S. An information-processing model of one kind of problem solving. Psychological Monographs, 1964, 78 (4, Whole No. 581).

Klausmeier, H. J., Harris, C. W., & Wiersma, W. Strategies of learning and efficiency of concept attainment by individuals and groups. U. S. Office of Education Cooperative Research Project No. 1442. Madison: University of Wisconsin, 1964.

Laughery, K. R., & Gregg, L. W. Simulation of human problem-solving behavior. Psychometrika, 1962, 27, 265-282.

Miller, G. A., Galanter, E., & Pribram, K. Plans and the structure of behavior. New York: Holt, 1960.

Newell, A. Some problems of basic organization in problem-solving programs. In M. Youitts, G. T. Jacobi, & A. D. Goldstein (Eds.), Self Organizing Systems. New York: Sparten, 1962.

Newell, A. (Ed.), Information processing language-V manual (2nd ed.) Englewood Cliffs: N.J.: Prentice-Hall, 1964.

Newell, A., Shaw, J. C., & Simon, H. A. Elements of a theory of human problem solving. Psychological Review, 1958, 65, 151-166.

Simon, H. A., & Kotovsky, K. Human acquisition of concept for sequential patterns. Psychological Review, 1963, 70, 534-546.

Uhr, L., & Vossler, C. A pattern recognition program that generates, evaluates, and adjusts its own operation. Proceedings of the Western Joint Computer Conference, 1961. Pp. 555-569.

APPENDIX
"THINK-ALOUD" PROTOCOL

Problem I (Concept is short neck, bent tail)

E: This card belongs to the concept. (Focus card is blue, s. neck, s. ears, b. tail)
S: Ah, lets see it is blue and has a short neck, and has a straight tail. Ah
E: This card.
S: This ?
E: Yea
S: Short neck, straight tail
E: No that tail is bent.
S: Oh wait!
E: See this has a straight tail. You can compare them.
S: Oh, that's bent and that's straight. Alright, I was looking at this, so I thought that this was the real bent one.
E: Oh, I'm sorry.
S: In other words these are the same categories.
E: Well I call them curly
S: So there is another category
E: There are three kinds of tails yea.
S: Let me see.
E: Straight, bent, and curly.
S: O.K., Ah
E: Or you can call them what ever you want to.
S: It has a bent tail, it's blue, short neck, short ears. Um. Let me see, uh
E: What are you looking for ?
S: I'm looking for the same thing in another color to see if color is one of the categories, one of the characteristics. Is this the one ? It's brown, has short neck, and bent tail. (brown, s. neck, s. ears, b. tail, varying only color)
E: Yes, that does belong.
S: So then it doesn't matter what color it is. Um, I'll find one in yellow to see if... I can see this one in yellow. Does that belong ? (yellow, s. neck, s. ears, b. tail, varying only color again)
E: Yes that belongs.
S: Well, three of them are the same exactly except they're in different colors, therefore, the one I'm looking for... it doesn't matter what color the one I'm looking for is. So the characteristics are, let me see, um... I'm going to find out if the tail has to be bent or not, so I'll take one that has a straight tail and no ears.
E: What are you looking for ?
S: I'm looking for, oh, here's one with a straight tail, and big ears. Oh wait, a straight tail and small ears. Does that fit in ? (brown, s. neck, s. ears, str. tail, varying color and tail)
E: No, that does not belong
S: So obviously the tail is the one, the characteristic that uh, rules that one out. Does this belong ? It has a long neck and a curly tail. (brown, l. neck, s. ears, curly tail, varying color, neck, and tail. But S should have learned that tail is relevant)
E: No that does not belong ? So now what are you thinking ?
S: I'm thinking that, well, something with a curly tail does not belong in the category. Is it possible that the card I'm looking for must have any color, must have a bent tail, and no neck, and short ears, or short neck and short ears.
E: Are you guessing at it now ?
S: Well Uh-hum actually I am.
E: Would you mind repeating it.
S: Bent tail, short neck, no ears, uh, short ears, and it doesn't matter what color it is.
E: No it is not correct.
S: Well I will try and rule out some other category. Does uh, does this belong ? Has all the characteristics that the first part had, but it has big ears. Does this belong ? (blue, s. neck, l. ears, b. tail, varying just ears.)
E: Yes it does belong.

S: Oh, so big ears are part of the category. Uh. Does this belong? (blue, s. neck, l. ears, curly tail, varying ears and tail. S doesn't seem too sure of tail).
E: Why are you asking?
S: Because it has big ears, and a curly tail.
E: No that does not belong.
S: Then the curly tail must be a characteristic that doesn't belong. Does this belong? Has a straight tail and big ears. (blue, l. neck, l. ears, str. tail, varying neck, ears, and tail)
E: No that does not belong
S: Then a straight tail does not belong. Well, then after searching out all the characteristics, I feel that the card I'm looking for must have any color, but it must have a bent tail, and a short neck, but it can have any kind of ears.
E: That is correct. Uh-hum, O.K.

Problem 2 (Concept ~~is~~ yellow, long neck, curly tail)
E: This card belongs to the concept. (Focus card is yellow, l. neck, s. ears, c. tail)
S: Yes, it has a long neck, and it's yellow, and it has a curled tail, and has no ears. Now I'm going to test for, color I think first. Uh. Here is the same thing in blue, long neck and curled tail, and small ears. Does that fit? (blue, l. neck, s. ears, c. tail, varying just color)
E: No that does not belong.
S: So obviously it has the same characteristic, but a different color, then the color rules it out. Does this belong? Oh, wait now I'm sorry.
E: What are you looking for?
S: A brown animal with long neck, small ears, and a curled tail. But I don't seem to find one. Here's one. Does that fit into the category? (brown, l. neck, s. ears, c. tail, varying color again. In all these problems, S always varied color first and checked it twice.)
E: No this does not
S: Well I tested for the two other colors shown on the chart therefore, I feel that color is a characteristic that rules an animal out, so the color of an animal must be yellow. Now I will check for the size of the neck. Uh. Here is the same card, only that it has a small neck, same animal. Does that fit in? (yellow, s. neck, s. ears, c. tail, varying only neck.)
E: No that does not belong.
S: Well then the size of the neck is another characteristic that I'm looking for. It must have a long neck. Uh. The same card with a long neck, and big ears. Wait now I'm sorry. Here, does that card belong? (yellow, l. neck, l. ears, c. tail, varying only ears.)
E: Yes, it does belong.
S: So it doesn't matter what size the ears are? Here is an animal. I'm looking for an animal with a short neck to see if it is the size of the neck. (But S just checked the neck) Oh here, well it has a curled tail. Oh does this animal fit in it has a bent tail? (yellow, s. neck, s. ears, bent tail, varying neck and tail. S is either not paying attention or he has a very short memory)
E: No this does not belong.
S: Does the
E: What did that tell you?
S: Well it told me that, oh wait, I haven't tested really for a bent neck. Does this card with the short neck and curled tail fit in. Yes it does. (yellow, s. neck, s. ears, c. tail, varying neck. Same as 3rd choice. It seems that S forgot the designation rather than forgot he chose it.)
E: No it doesn't.
S: Oh, it doesn't!
E: Did you forget this?
S: Yes I forgot that. So the size of the neck does matter. Uh. I'm looking for, oh here's one. An animal with no ears, a long neck and a bent tail. Does that belong? (yellow, l. neck, s. ears, b. tail, varying just tail)
E: No it doesn't belong.
S: Well that tells me that the bent tail is out. Now I'm looking for an animal who has a long neck, and a curled tail, and no ears. Is there one? Oh, it's the only one there. I think that I have found it. The animal must be yellow, must have the curled tail, must have a long neck, and must have no ears, or small ears.
E: Short ears?
S: Short ears.
E: No that's not correct.
S: Oh wait, I think that I tested for the ears. Then it must have all the characteristics, but it doesn't matter what ears.
E: O.K. (Laughed) That's better.

Problem 3 (Concept is brown, short ears)
E: This card belongs to the concept. (Focus card is brown, l. neck, s. ears, b. tail)

S: It is a brown animal, it has a long neck, it has short ears, and a bent tail. Uh, I would like to test for color first, so I will find the same animal in a different color, and, see, long neck, bent tail, and no ears, um. Does this animal fit in? (blue, l. neck, s. ears, b. tail, varying color)

E: Uh. No this does not belong.

S: Well obviously the blue animal with the same characteristics doesn't fit in, so I will look for a brown animal, and see if that fits in. The same characteristics. Oh I mean a yellow animal. Does this animal fit in? (yellow, l. neck, s. ears, b. tail, varying color again)

E: No it does not belong.

S: Well that tells me that color is a characteristic that rules an animal out. Now I will look for the same animal with a straight tail. Does this animal fit in, the same color, but a straight tail? (brown, l. neck, s. ears, str. tail, varying only tail)

E: Which one? Yes that does belong.

S: So it does not matter if the tail is bent or straight. I will find one with a short neck. Does this animal fit in? (brown, s. neck, s. ears, str. tail, varying neck and tail)

E: Yes that does belong.

S: It doesn't matter if the neck is short or tall, but this animal does have short ears. I will find one with big ears. Does this animal fit in? (brown, l. neck, l. ears, str. tail, varying ears and tail)

E: No it does not.

S: Well I feel that the animal must be brown, and must have long or short neck, and must ...Oh, I haven't tested for a curled tail yet. Does this animal fit in? (brown, s. neck, s. ears, curly tail, varying neck and tail again)

E: Yes it does belong.

S: Well then the animal must be brown, must have a long or short neck. Must have, well it doesn't matter what size neck, or what kind of tail it has, but it must have small ears.

E: O.K., that is correct. Um-hum. (S follows a conservative strategy and varies all the values of a 3-valued dimension)

Problem 4 (Concept is brown, short neck straight tail)

E: This card belongs to the concept. What is the first thing you think of when I point a card out to you? (Focus card is brown, s. neck, s. ears, str. tail)

S: What do you mean the characteristics, or just the first...

E: Well the first thing you think of.

S: I think of a dachshund. It looks like a dachshund.

E: (Laughed) NO, I mean, you know,

S: Brown is the first thing, and the fact that it has a short neck, and no ears, so I think that is pretty important, and a straight tail and no other characteristics. Uh.

E: Then what do you think? Continue

S: I'm trying to decide, I'm looking for a card that's the exact same thing but a different color, and here is one in blue. (blue, s. neck, s. ears, str. tail)

E: No this does not belong.

S: Well I will find one in yellow and see if yellow belongs.

E: You usually test color first?

S: Yes I do. I think that is a good way to start anyway. I guess it really doesn't matter, it is the easiest, I feel, if you differentiate in color because then you can look for other characteristics. Does this animal, yellow one fit in? (yellow, s. neck, s. ears, str. tail)

E: No it does not belong.

S: That tells me that the animal that I'm looking for must be brown, so it narrows down the field, and it is very easy to find animals by color rather than other characteristics, which aren't as visible. Uh, I'm checking the tail. Does this animal fit in? Oh wait, I'm sorry. I'm looking for a short neck. Does this animal fit in? (brown, s. neck, s. ears, b. tail, varying just tail)

E: No it does not belong.

S: Tells me that the tail must be straight. Does this animal fit in? The straight tail and long neck. (brown, l. neck, s. ears, str. tail, varying just neck)

E: What do you want to find out?

S: The neck.

E: No that does not belong

S: It tells me that the neck must be short. Does this animal fit in? It has big ears. (brown, s. neck. l. ears, str. tail, varying just ears)

E: Yes it does belong

S: Well that tells me that the animal must be brown, must have a straight tail, must have a short neck, and big ears. (S has just described the last card he has chosen.)

E: And big ears?

S: No, small ears. Oh wait it doesn't matter what ears.

E: O.K. That is correct.

Problem 5 (Concept is short ears)

E: This card belongs to the concept. (Focus card is blue, s. neck, s. ears, curly tail)

S: It is a blue animal with a curled tail, short neck, no ears. I'm going to check the color first so it is the same animal but a different color. (brown, s. neck, s. ears, c. tail, varying color)

E: Uh-hum. Yet it does belong.

S: Brown belongs, I'll see if yellow belongs too. (yellow, s. neck, s. ears, c. tail)

E: Yes that does belong

S: Then that tells me that it doesn't matter what color it is. I will check for the tail first. Does this animal fit in? (yellow, s. neck, s. ears, b. tail, varying color and tail)

E: Yes that belongs too.

S: Well then that tells me that it doesn't matter if the tail is bent. Does this animal fit in? Let me see I'm looking for a curled tail. Oh wait that is a curled tail, I'm sorry, I'm looking for a small animal with a straight tail. Here it is. Does this animal fit in? (blue, s. neck, s. ears, str. tail, varying tail. Usually after checking both other values of color, <u>S</u> checks both other values of tail)

E: Yes it does belong.

S: So that tells me that it doesn't matter what color it is or what kind of tail it has. Now I'm checking for a long neck. Does this animal fit in? Has a long neck. (yellow, l. neck, s. ears, c. tail)

E: Yes, it does belong.

S: Then that tells me that it doesn't matter what neck it has. Does this animal fit in? (yellow, s. neck, l. ears, str. tail)

E: No that does not belong.

S: It has the characteristics that are acceptable. But it has big ears, and I haven't checked for that yet.

E: No that does not belong.

S: That tells me that the animal may be any color, and that it may have any size neck, may have any kind of tail, but it must have small ears.

E: That is correct.

Pro[illegible] 6 (Concept is curly tail)

E: That card belongs to the concept. (Focus card is brown, short neck, s. ears, c. tail)

S: That one.

E: Yea.

S: Uh. It has a short neck, small ears, and a curled tail, and it's brown. I will find the same thing in a different color. Lets see. Wait. Does this fit in? (blue, s. neck, s. ears, c. tail)

E: Yes it does belong.

S: Does this fit? (yellow, s. neck, s. ears, c. tail)

E: Yes it does.

S: That tells me that it doesn't matter what color it is. I'm looking for the size of the neck now. Does this fit in? (yellow, l. neck, s. ears, c. tail, varying color and neck.)

E: Yes it does belong.

S: That tells me that it doesn't matter what size the neck is. Does this fit in? (yellow, s. neck, l. ears, str. tail, varying color, ears, and tail)

E: No that does not belong

S: Ah wait that doesn't tell me anything because there are two different characteristics, I should have pointed to something else. Does this fit in? Oh wait, yea. Does this fit in? (blue, s. neck, l. ears, c. tail, varying color and ears)

E: What are you trying to find out?

S: What size the ears are.

E: Yea that belongs.

S: That tells me that it doesn't matter what size the ears are. Does this fit in? (blue, l. neck, l. ears, c. tail, varying everything except tail)

E: Yes that belongs.

S: Well thats for the long neck. Now I have tested for everything else? Does this fit in? (blue, l. neck, l. ears, b. tail, varying everything. <u>S</u> should have known the concept 2 choices ago.)

E: Which one, the blue one?

S: Yea.

E: No that does not belong.

S: Does this fit in, with the straight tail. (blue, s. neck, s. ears, str. tail, varying color and tail)

E: No that does not belong.

S: That tells me that I'm looking for an animal that can be of any color, can have any size neck or any size ears, but must have a curled tail.

E: That is correct. O.K. Tell me what your impression of this board is and the procedure and everything. Any thing you would like to comment about.

S: Uh-hum. I think it is a good way to test, well first of all the colors are good because I think you can differentiate between them pretty easily, and so that is the immediate stimulus I guess. The immediate thing that I see is a difference in color the first thing, and then if you distinguish between the colors first then you can find the different characteristics. It can be sort of confusing, forgetting if it has a bent tail, straight. I really don't see the purpose of it all. Unless it is an I. Q. or something.

E: Well this is just to see the little detailed thought processes that is behind solving problems like this.

S: Uh-hum.

E: That is all that you have to say on the subject?

S: Yea I think so.

E: What is your general strategy of solving the problem?

S: Well first I try to solve the problem with color. I think that it is easiest to differentiate, between colors, and then I usually look for neck first, and then, because that is another easy way to differentiate because it is outstanding, and then for the tail, and the ears, it doesn't really matter which way you go about it because they are both equally as easy to see.

E: Uh-hum. O.K. That will be all for today. (S always picked color first. Then he would pick tail, not neck, in 2/3 of the problems. After that, neck, then ears. This S had been run initially on the old board of circles and triangles on colored paper. That is where he developed his conservative strategy. I ended this session early because S was getting very bored by that time.)